NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

TRACKING MULTIPLE TARGETS IN CLUTTERED ENVIRONMENTS WITH THE PROBABILISTIC MULTI-HYPOTHESIS TRACKING FILTER

by

Darin T. Dunham

March 1997

Thesis Advisor: Robert G. Hutchins

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 1997	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE TRACKING MULTIPLE TARGETS IN CLUTTERED ENVIRONMENTS WITH THE PROBABILISTIC MULTI-HYPOTHESIS TRACKING FILTER	5. FUNDING NUMBERS
6. AUTHOR(S) Darin T. Dunham	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

Tracking multiple targets in a cluttered environment is extremely difficult. Traditional approaches generally use simple techniques that combine gating with some form of nearest neighbor association to reduce the effects of clutter. When clutter densities increase, these traditional algorithms fail to perform well. To counter this problem, the multi-hypothesis tracking (MHT) algorithm was developed. This approach enumerates almost every conceivable combination of measurements to determine the most likely tracks. This process quickly becomes very complex and requires vast amounts of memory in order to store all of the possible tracks.

To avoid this complexity, more sophisticated single hypothesis data association techniques have been developed, such as the probabilistic data association filter (PDAF). These algorithms have enjoyed some success, but do not take advantage of any future data to help clarify ambiguous situations.

On the other hand, the probabilistic multi-hypothesis tracking (PMHT) algorithm, proposed by Streit and Luginbuhl in 1995, attempts to use the best aspects of the MHT and the PDAF. In the PMHT algorithm, data is processed in batches, thereby using information from before and after each measurement to determine the likelihood of each measurement-to-track association. Furthermore, like the PDAF, it does not attempt to make hard assignments or enumerate all possible combinations, but instead associates each measurement with each track based upon its probability of association.

Actual performance and initialization of the PMHT algorithm in the presence of significant clutter has not been adequately researched. This study focuses on the performance of the PMHT algorithm in dense clutter and the initialization thereof. In addition, the effectiveness of measurement attribute data is analyzed, especially as it relates to algorithm initialization. Further, it compares the performance of this algorithm to the nearest neighbor, MHT, and PDAF.

14. SUBJECT TERMS Probabilistic; Tracking; Clutter; Multiple Targets; Kalman Filter	15. NUMBER OF PAGES 87
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

Approved for public release; distribution is unlimited

TRACKING MULTIPLE TARGETS IN CLUTTERED ENVIRONMENTS WITH THE PROBABILISTIC MULTI-HYPOTHESIS TRACKING FILTER

Darin T. Dunham
Captain, United States Marine Corps
B.S., Carnegie Mellon University, 1991

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 1997

ABSTRACT

Tracking multiple targets in a cluttered environment is extremely difficult. Traditional approaches generally use simple techniques that combine gating with some form of nearest neighbor association to reduce the effects of clutter. When clutter densities increase, these traditional algorithms fail to perform well. To counter this problem, the multi-hypothesis tracking (MHT) algorithm was developed. This approach enumerates almost every conceivable combination of measurements to determine the most likely tracks. This process quickly becomes very complex and requires vast amounts of memory in order to store all of the possible tracks.

To avoid this complexity, more sophisticated single hypothesis data association techniques have been developed, such as the probabilistic data association filter (PDAF). These algorithms have enjoyed some success, but do not take advantage of any future data to help clarify ambiguous situations.

On the other hand, the probabilistic multi-hypothesis tracking (PMHT) algorithm, proposed by Streit and Luginbuhl in 1995, attempts to use the best aspects of the MHT and the PDAF. In the PMHT algorithm, data is processed in batches, thereby using information from before and after each measurement to determine the likelihood of each measurement-to-track association. Furthermore, like the PDAF, it does not attempt to make hard assignments or enumerate all possible combinations, but instead associates each measurement with each track based upon its probability of association.

Actual performance and initialization of the PMHT algorithm in the presence of significant clutter has not been adequately researched. This study focuses on the performance of the PMHT algorithm in dense clutter and the initialization thereof. In addition, the effectiveness of measurement attribute data is analyzed, especially as it relates to algorithm initialization. Further, it compares the performance of this algorithm to the nearest neighbor, MHT, and PDAF.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS AND SYMBOLS

$\mathbf{C}_{tm}$	Measurement matrix for target m at time t
$\mathbf{G}_t$	Kalman gain matrix at time t
i	Index of PMHT algorithm
M	Number of targets (target motion models)
MHT	Multi-hypothesis tracking
n_t	Number of measurements in the scan at time t
$\mathbf{P}_{tm}$	Target state covariance matrix for target m at time t
PDAF	Probabilistic data association filter
PMHT	Probabilistic multi-hypothesis tracking
$\mathbf{Q}_{tm}$	Process covariance matrix for target m at time t
r	Index used for measurements within a scan
$\mathbf{R}_{tm}$	Noise covariance matrix for target m at time t
s,m	Indices used for target states
T	Batch length (number of scans processed together)
t	Index used for discrete time
$\mathbf{X}_m = (x_{0m},\ldots,x_{Tm})$	Vector of all target state random variables for target m
$\mathbf{X}_t = (x_{t1},\ldots,x_{tM})$	Vector of all target state random variables at time t
x_{tm}	State of target m at time t, where t=0,…,T and m=1,…,M
w_{mtr}	Probability of measurement r originating from target m at time t
$\mathbf{Z}_t = (z_{t1},\ldots,z_{tn_t})$	Scan measurement vector at time t
z_{tr}	Measurement r in scan t
$\pi_m = (\pi_{1m},\ldots,\pi_{Tm})$	Measurement probability vector for target m
$\pi_t = (\pi_{t1},\ldots,\pi_{tM})$	Target measurement probability vector for scan t
π_{tm}	Probability of measurements in scan t originating from target m
ρ	Clutter weight constant
Σ_{tm}	Weighting matrix for target m at time t
Φ	Transition matrix

ACKNOWLEDGMENT

I would like to thank my thesis advisor, Gary Hutchins, for all of his advice, guidance, and availability during this endeavor.

Furthermore, I would especially like to thank my wife, Martha, for her support and input. Finally, I dedicate this thesis to my two children, Nathan and Amy.

I. INTRODUCTION

A. BACKGROUND

Tracking multiple targets in a cluttered environment, such as is found in littoral waters, is an extremely difficult task. This is due to the added noise which is caused by the closeness of the ocean floor to the surface. Furthermore, in littoral waters, there are quite often various man-made structures, which can cause the addition of false target sonar returns. In this environment, targets typically operate at speeds between 2-10 knots. In order to maintain steerage control, submarines find it difficult to operate below this minimum velocity, and above 10 knots, diesel submarines will produce an obvious amount of noise.

Traditional approaches used to solve the cluttered environment tracking problem have typically employed simple techniques to determine what is a true measurement from a target and what is not. They accomplish this by a combination of gating (discarding measurements) and some form of nearest neighbor association (picking the closest measurement to the current target position estimate, where "closest" is defined by a weighted distance). In the open ocean, where the water is deep, traditional approaches perform well because the problem of dense clutter is not encountered. However, in littoral waters, where clutter densities increase, these traditional algorithms fail to yield reliable results. In order to solve the clutter problem, two algorithms were developed—

the multi-hypothesis tracking (MHT) algorithm and the probabilistic data association filter (PDAF).

1. The Multi-Hypothesis Tracker (MHT)

The MHT [Ref. 5] enumerates almost all of the possible combinations of measurement-to-track assignments. Then from all of these possibilities, the most likely is selected as the best estimate of the track. The problem quickly becomes extremely complex as the data combinations are growing exponentially as each new measurement batch is received. This requires a huge amount of memory and computing power. Furthermore, as the number of possibilities increase, some form of pruning must be done in order to keep the number of hypotheses within limits.

2. The Probabilistic Data Association Filter (PDAF)

On the other hand, the PDAF [Ref. 4] does not make hard measurement-to-track assignments, but rather weights each measurement based upon its likelihood of association with a track. This algorithm has some advantages in its simplicity, especially in computational and storage costs. However, it only gets one chance to weight the data correctly. Therefore, this algorithm does not take advantage of any future data before making a decision on the most likely true measurement.

B. A NEW APPROACH

In 1995, Streit and Luginbuhl of the Naval Undersea Warfare Center proposed a new algorithm called the probabilistic multi-hypothesis tracking (PMHT) algorithm [Ref. 1]. Like the MHT, this algorithm processes data in batches, thereby giving it the

advantage of future data before decisions are made. However, the PMHT does not attempt to enumerate all possible combinations, but rather weights the measurements based on the likelihood of each measurement being the true measurement. Therefore, like the PDAF, this new algorithm employs an empirical, Bayesian data association to score the measurements in order to determine the likely true measurement centroid. This technique can be significantly faster than the MHT, but will require more time to compute than the PDAF. This research focuses on two primary areas of the PMHT that have yet to be studied carefully, that is, the actual performance of the algorithm in the presence of dense clutter and the initialization thereof.

Furthermore, this thesis also addresses the performance of the PMHT as compared to the traditional tracking algorithms—the MHT and PDAF. In addition, measurement attribute data is explored in conjunction with the PMHT algorithm.

C. THESIS OUTLINE

This thesis is divided into the following chapters: Chapter II describes the theory and derivation behind the PMHT algorithm. Chapter III lays out the explicit algorithm as it has been implemented in this study. Chapter IV covers the difficulty of initializing this algorithm in the presence of clutter, and how the initialization was eventually accomplished. Chapter V shows the results of this implementation of the PMHT algorithm and compares these results to other tracking algorithms, i.e., the nearest neighbor, PDAF, and MHT. Portions of these results were published and presented at the Asilomar Conference on Signals, Systems, and Computers in November of 1996 [Ref. 2].

Finally, Chapter VI summarizes this study and offers suggestions about areas in which further research might be conducted.

II. THEORETICAL BASIS OF THE PMHT ALGORITHM

A. TARGET MOTION MODEL

In this research, measurements are obtained from a sensor in batches at a set time interval. The PMHT algorithm takes all of the received measurements and computes an optimal estimate for each target track. The targets are assumed to be independent with linear Gaussian statistics of the following form:

$$\mathbf{x}_{t+\Delta t} = \Phi \mathbf{x}_t + \mathbf{w}_t \tag{2.1}$$

where $\mathbf{x}$ is the state-space vector containing the x position, x velocity, y position, and y velocity, respectively. Φ is the following discrete state-space matrix:

$$\Phi = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

Further, $\mathbf{w}_t$ is white Gaussian noise with known covariance matrix $\mathbf{Q}$, which is given by:

$$\mathbf{Q} = q \begin{bmatrix} \Delta t^3/3 & \Delta t^2/2 & 0 & 0 \\ \Delta t^2/2 & \Delta t & 0 & 0 \\ 0 & 0 & \Delta t^3/3 & \Delta t^2/2 \\ 0 & 0 & \Delta t^2/2 & \Delta t \end{bmatrix} \tag{2.3}$$

where Δt is the time in between scans, and q is a parameter which reflects the maneuvering behavior of the target. This parameter is used to adjust the performance of the Kalman Filter. Usually for fairly straight tracks, q is set to a small but nonzero value in order to prevent covariance collapse in the Kalman algorithm.

B. MEASUREMENT MODEL

The measurement model in this research assumes that a sensor returns range and bearing information which contains additive Gaussian noise. Therefore, each measurement pair is of the following form:

$$\mathbf{z}_{rt} = \begin{bmatrix} r_{rt} \\ \phi_{rt} \end{bmatrix} + \mathbf{e}_t \tag{2.4}$$

where the subscript r denotes the index for measurements within a scan ($r = 1,\ldots,n_t$), and the subscript t specifies the discrete time index ($t = 0,\ldots,T$). Hence, there are n_t total measurements taken at time t, and there are T total scan times in the scenario. The error vector is additive Gaussian noise with zero mean and covariance given by:

$$\mathbf{R} = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\phi^2 \end{bmatrix} \tag{2.5}$$

For this research, $\sigma_r = 100$ meters and $\sigma_\phi = 3$ degrees was assumed. In this case, with the coordinates in polar form, the measurement covariance matrix is the same for all measurements at all time scans. However, if the measurements were in another coordinate system (e.g., Cartesian) then each measurement would have its own covariance matrix.

C. COORDINATE CONVERSION

The Kalman Smoother requires a linear state equation and a linear measurement equation. The bearing measurement in polar coordinates is nonlinear. Therefore, in order to use these measurements in the classical Kalman Smoother, it is necessary to convert

them into Cartesian coordinates. Lerro and Bar-Shalom have demonstrated that converting range-bearing measurements into Cartesian space prior to implementing the Kalman algorithm is superior to utilizing the raw range-bearing measurements directly in the Extended Kalman algorithm [Ref. 3].

Lerro and Bar-Shalom recommend converting the measurements to Cartesian coordinates using their "debiased" equations. These equations convert both the measurement itself and its associated covariance matrix. The following are the debiased measurement conversions:

$$\mathbf{z}_{rt} = \begin{bmatrix} x_{rt} \\ y_{rt} \end{bmatrix} = \begin{bmatrix} r_{rt}\cos(\phi_{rt})[1-(e^{-\sigma_\phi^2} - e^{-\sigma_\phi^2/2})] \\ r_{rt}\sin(\phi_{rt})[1-(e^{-\sigma_\phi^2} - e^{-\sigma_\phi^2/2})] \end{bmatrix} \tag{2.6}$$

Furthermore, the corresponding covariance matrix is given by:

$$\mathbf{R}_{rt} = \begin{bmatrix} R_{11} & R_{12} \\ R_{12} & R_{22} \end{bmatrix} \tag{2.7}$$

where,

$$\begin{aligned}
R_{11} &= r_{rt}^2 e^{-2\sigma_\phi^2}[\cos^2\phi_{rt}(\cosh 2\sigma_\phi^2 - \cosh\sigma_\phi^2) + \sin^2\phi_{rt}(\sinh 2\sigma_\phi^2 - \sinh\sigma_\phi^2)] \\
&\quad + \sigma_r^2 e^{-2\sigma_\phi^2}[\cos^2\phi_{rt}(2\cosh 2\sigma_\phi^2 - \cosh\sigma_\phi^2) + \sin^2\phi_{rt}(2\sinh 2\sigma_\phi^2 - \sinh\sigma_\phi^2)] \\
R_{12} &= \sin\phi_{rt}\cos\phi_{rt} e^{-4\sigma_\phi^2}[\sigma_r^2 + (r_{rt}^2 + \sigma_r^2)(1 - e^{\sigma_\phi^2})] \\
R_{22} &= r_{rt}^2 e^{-2\sigma_\phi^2}[\sin^2\phi_{rt}(\cosh 2\sigma_\phi^2 - \cosh\sigma_\phi^2) + \cos^2\phi_{rt}(\sinh 2\sigma_\phi^2 - \sinh\sigma_\phi^2)] \\
&\quad + \sigma_r^2 e^{-2\sigma_\phi^2}[\sin^2\phi_{rt}(2\cosh 2\sigma_\phi^2 - \cosh\sigma_\phi^2) + \cos^2\phi_{rt}(2\sinh 2\sigma_\phi^2 - \sinh\sigma_\phi^2)]
\end{aligned}$$

With these equations, there is a different covariance matrix for each measurement at each different time scan.

D. ITERATIONS

Each iteration of the PMHT algorithm begins with a set of track position estimates and a set of measurement probabilities. Then the weights are computed and centroids formed. These centroids are then used in the Kalman Smoother to update the track position estimates. With the new estimates, a new set of weights is computed. These weights will be similar to the previous weights, but they will be different because of the new estimates. After some iterations, the weights will converge. When convergence is reached, the current estimate is theoretically the optimal estimate for a given track.

E. WEIGHTING THE MEASUREMENTS

In this research, two different models were used to assign weights to measurements. The first is the target track model, which uses a normal distribution to assign weights to measurements. The second is the clutter model, which uses a constant value to assign weights to the measurements.

1. Track Model

Given the measurements and an initial estimate at each time scan, the PMHT uses a normal distribution between the estimate and each measurement to determine the value of the weight assigned to each measurement in a given scan. This weight specifies the likelihood that this measurement belongs to a particular track model. From these weights and measurements, a centroid is computed for each track model. The centroid is calculated by simply multiplying each measurement by its associated weight and then summing all of these together.

The weight calculated for each measurement is conditioned on the position of the estimate and its covariance matrix. If a given measurement is far from the estimate, it will get a low weight. On the other hand, if a measurement is near the estimate, it will receive a high weight.

2. Clutter Model

Since uniform clutter is assumed for this research, the clutter weight is equal to a constant, which can be adjusted. The clutter weight was initially determined by calculating the area of interest for which measurements will be returned. The inverse of this area was then used as the starting point for the clutter weight value. For optimal performance, a clutter weight value an order of magnitude less than this value was used.

F. KALMAN SMOOTHING

The estimates at each scan are linked together by the Kalman Smoother. The centroids at each time scan are used as the "measurements" in the Kalman Smoother. This produces a new set of estimates for each track model. The Kalman Smoother is used so that all estimates are updated using all the available information from before and after each time scan. This produces the best estimate possible at each time scan.

If the conventional Kalman is used, then each measurement will have its own corresponding covariance matrix. This covariance matrix is used both in the Kalman Smoothing step, as well as in the weighting step. This not only complicates the computations and requires more memory storage, but also causes both the position estimate and its associated covariance matrix to have to converge in the iteration process.

On the other hand, if the Extended Kalman is used, the same covariance matrix is used for all measurements at all time increments not only in computing the weights, but also in the smoothing process. This allows simplification of the computations, and during the iteration process, only the position estimate is being refined and convergence is more easily achieved.

III. THE PMHT ALGORITHM IN EXPLICIT FORM

A. THE BASIC ALGORITHM

This algorithm is taken from Streit and Luginbuhl [Ref. 1] using the linear Gaussian case. In section B, I will discuss the modifications which were made to the original algorithm, initially. Then in section C, I will cover the modifications, which further improved the performance of the algorithm.

1. Initialization

Measurement probabilities, $\Pi^{(0)} = \left\{\pi_{tm}^{(0)}\right\}$ must be assigned so that $\pi_{tm}^{(0)} > 0$. It is not critical what values are assigned to these measurement probabilities because in the first iteration they will be recalculated. Moreover, they do not have an adverse effect before they are recomputed. The $\pi_{tm}^{(i)}$ values specify the estimated probability that a measurement at scan t is assigned to target model *m* after *i* iterations of the PMHT algorithm.

An initial target state $\left(x_{0m}^{(0)}, x_{1m}^{(0)}, \ldots, x_{Tm}^{(0)}\right)$ for each time increment and each of the *M* target models must be assigned. My experience has shown that these initial estimates must be fairly accurate to ensure that the algorithm performs satisfactorily as clutter densities increase.

In this paper, *m* specifies the target model ($m = 1,\ldots,M$); *t* specifies the discrete time index ($t = 0,\ldots,T$); *r* specifies the index for measurements within a scan ($r = 1,\ldots,n_t$); and the superscript *i* specifies the iteration index ($i = 0,1,\ldots$).

2. Computation of the Weights

For every target and measurement combination at each scan, a weight is computed. The value of the likelihood function (assuming normal distribution) evaluated at the error between the current estimated position and each measurement is used for the weight:

$$w_{mtr}^{*(i+1)} = \frac{\exp\left(-\frac{1}{2}\tilde{\mathbf{z}}_{mtr}^{(i)\mathrm{T}}\Sigma_{tm}^{-1}\tilde{\mathbf{z}}_{mtr}^{(i)}\right)}{2\pi\sqrt{\det(\Sigma_{tm})}} \tag{3.1}$$

$$w_{mtr}^{(i+1)} = \frac{w_{mtr}^{*(i+1)}}{\sum_{s=1}^{M}\pi_{ts}^{(i)}w_{str}^{*(i+1)}} \tag{3.2}$$

where,

$$\tilde{\mathbf{z}}_{mtr}^{(i)} = \mathbf{z}_{tr} - \hat{\mathbf{z}}_{mtr}^{(i)} = \mathbf{z}_{tr} - \mathbf{C}_{tm}\mathbf{x}_{tm}^{(i)} \tag{3.3}$$

is the error between the current estimate and a measurement. Further, Σ is the weighting matrix defined as:

$$\Sigma_{tm} = \mathbf{C}_{tm}\mathbf{P}_{tm}\mathbf{C}_{tm}^{\mathrm{T}} + \mathbf{R}_{tm} \tag{3.4}$$

Here $\mathbf{P}_{tm}$ is the covariance matrix associated with the target state estimate, and $\mathbf{C}_{tm}$ is the measurement matrix defined as:

$$\mathbf{C}_{tm} = \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.5}$$

for the standard Kalman algorithm.

3. Calculation of the Measurement Centroids

First, the mean measurement weight for each target model m at time t is defined as:

$$\overline{w}_{mt}^{(i+1)} = \frac{1}{n_t} \sum_{r=1}^{n_t} w_{mtr}^{(i+1)} \tag{3.6}$$

Next, the measurement centroid is computed as:

$$\overline{\mathbf{z}}_{tm}^{(i+1)} = \frac{1}{n_t \overline{w}_{tm}^{(i+1)}} \sum_{r=1}^{n_t} w_{mtr}^{(i+1)} \mathbf{z}_{tr} \tag{3.7}$$

This measurement centroid will be used in the Kalman Smoothing step below.

4. Target Measurement Probabilities Update

The next step before the Kalman Smoother is to update the target measurement probabilites. This is computed as:

$$\pi_{tm}^{(i+1)} = \overline{w}_{mt}^{(i+1)} \pi_{tm}^{(i)} \tag{3.8}$$

5. Target State Sequences Update

The target state sequences are updated via the Kalman Smoother using the measurement centroids as the inputs. First, the intermediate variables of the forward recursion are initialized as:

$$\hat{\mathbf{y}}_{0|0} = \mathbf{x}_{0m}^{(i)} \tag{3.9}$$

$$\mathbf{P}_{0|0} = \mathbf{P}_{0m} \tag{3.10}$$

Here with these dummy variables, the model m and iteration index i have been suppressed for notational simplicity. The forward recursion is defined for $t = 0,1,...,T\text{-}1$ as:

$$\mathbf{P}_{t+1|t} = \Phi \mathbf{P}_{t+1|t} \Phi^{\mathrm{T}} + \mathbf{Q} \tag{3.11}$$

$$\mathbf{G}_{t+1} = n_{t+1} \pi_{t+1,m}^{(i+1)} \mathbf{P}_{t+1|t} \mathbf{C}_{t+1,m}^{\mathrm{T}} \left[n_{t+1} \pi_{t+1,m}^{(i+1)} \mathbf{C}_{t+1,m} \mathbf{P}_{t+1|t} \mathbf{C}_{t+1,m}^{\mathrm{T}} + \mathbf{R}_{t+1,m} \right]^{-1} \tag{3.12}$$

$$\mathbf{P}_{t+1|t+1} = \mathbf{P}_{t+1|t} - \mathbf{G}_{t+1} \mathbf{C}_{t+1,m} \mathbf{P}_{t+1|t} \tag{3.13}$$

$$\hat{\mathbf{y}}_{t+1|t+1} = \Phi \hat{\mathbf{y}}_{t|t} + \mathbf{G}_{t+1} \left[\bar{\mathbf{z}}_{t+1,m}^{(i+1)} - \mathbf{C}_{t+1,m} \Phi \hat{\mathbf{y}}_{t|t} \right] \tag{3.14}$$

Then the updated target state estimate for model m at time t is:

$$\mathbf{x}_{Tm}^{(i+1)} = \hat{\mathbf{y}}_{T|T} \tag{3.15}$$

and the updated target state estimates for $t = T\text{-}1,...,1,0$ are computed via the backward recursion as:

$$\mathbf{x}_{tm}^{(i+1)} = \hat{\mathbf{y}}_{t|t} + \mathbf{P}_{t|t} \Phi^{\mathrm{T}} \mathbf{P}_{t+1|t}^{-1} \left[\mathbf{x}_{t+1,m}^{(i+1)} - \Phi \hat{\mathbf{y}}_{t|t} \right] \tag{3.16}$$

The equations in this subsection make up a bank of M Kalman Smoothers which can be run in parallel, although these filters are not independent because they are linked by the weights. In these equations, Φ is the same as was defined in (Eqn. 2.2).

Therefore, a block diagram of the basic algorithm is shown below in Figure 3.1. The convergence block will be discussed in the next section.

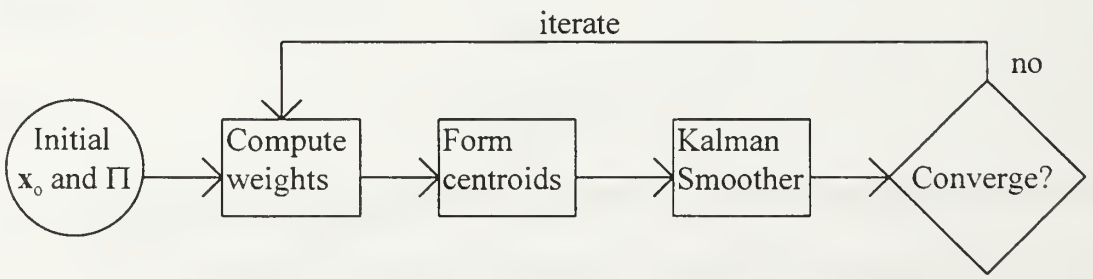

Figure 3.1. Basic PMHT

B. INITIAL MODIFICATIONS

Several modifications and additions are necessary in order for the PMHT algorithm to begin working.

1. Clutter Weight Model

First, there needs to be a model for clutter weights. Since uniform clutter is assumed for this problem, I used a clutter weight equal to a constant, which could be adjusted. Therefore,

$$w_{mlr}^{*(i+1)} = \rho \tag{3.17}$$

was used for the clutter model and (Eqn. 3.1) was used for target track models.

2. Convergence Criteria

The basic algorithm does not specifically state how convergence is to be determined. In this research, convergence was measured by the π values. It would also be possible to test convergence through the weights. However, both approaches yield similar results and the π values are quicker to sum and compare. Therefore, initially convergence was achieved when

$$\sum_{t=0}^{T}\sum_{m=1}^{M}\left|\pi_{tm}^{(i)} - \pi_{tm}^{(i-1)}\right| < Kc > 0 \tag{3.18}$$

The parameter Kc is adjusted for optimal performance. If this number is set too high, then the algorithm will stop before it has fully converged to its best solution. On the other hand, if this number is set too low, then the algorithm might never be able to meet this criteria. Initially Kc was set to 10^{-4}. Iterations are allowed to continue until

convergence is reached or the maximum limit is exceeded (100 iterations). I will refer to this convergence parameter as the stopping criteria in the sections below.

3. Measurement Covariance

Using the basic algorithm in Cartesian Coordinates requires that each measurement have its own associated covariance matrix. For the weighting equation (Eqn. 3.1), it is possible to use each measurement's associated covariance matrix. However, during the Kalman Smoothing step, a covariance matrix is needed for each track at each time scan. Furthermore, since the measurement centroids are mixtures of the received measurements, there is not just one matrix for each track model at each scan. The following variations were investigated during this research:

a. Weighted Covariance

For this calculation, a weighted covariance matrix is obtained by a similar process as is used to obtain the measurement centroids.

$$\tilde{\mathbf{R}}_{lm}^{(i)} = \sum_{r=1}^{n_t} \frac{w_{mtr}^{(i)}}{n_t \overline{w}_{mtr}^{(i)}} \mathbf{R}_{tr} \tag{3.19}$$

b. Closest Measurement

The covariance matrix associated with the measurement which is closest to the current estimate is used.

$$\hat{\mathbf{R}}_{lm}^{(i)} = \mathbf{R}_{tr} \tag{3.20}$$

c. Covariance of the Estimate

Here, the covariance matrix is computed from the current estimated position by using the debiased equations (Eqn. 2.7).

$$\hat{\mathbf{R}}_{tm}^{(i)} = \mathbf{R}(\mathbf{x}_{tm}^{(i)}) \tag{3.21}$$

Of these variations, the estimated covariance (Eqn. 3.21) has shown the most robustness and has provided the best overall results. Furthermore, it has been found that using this estimated covariance matrix not only in the Kalman Smoother (Eqn. 3.12), but also in the computation of the weights (Eqn. 3.4), provided even better results.

4. Distant Clutter

It was found that if a measurement point was too far from the estimated position then the weight calculated was extremely small, and numerical instabilities would be encountered. Therefore, on a suggestion from Dr. Streit, any measurements beyond a Chi-squared cut-off value of 0.995 from the estimated track position have their weight set to a low constant value of 10^{-20}, rather than using the actual weight from (Eqn. 3.1).

5. Five Scan Batches

The initialization process (to be discussed in Chapter IV) produces the initial estimates for the first five time scans. The PMHT algorithm is run on these five time scans until convergence is reached. Then the next five time scans are predicted, and the algorithm runs over the now 10 time scans, and so forth. This not only provides a more realistic approach to what might actually be implemented, but also allows the algorithm to only have to sort out five new points at a time.

6. Speed Limit

Since the targets of concern in this research have speeds of 2-10 knots, a 10 knot maximum speed is imposed when predicting ahead. The actual target estimates produced

by the PMHT algorithm are not limited and can converge to an estimate with any velocity.

Therefore, a block diagram with these modifications is shown in Figure 3.2.

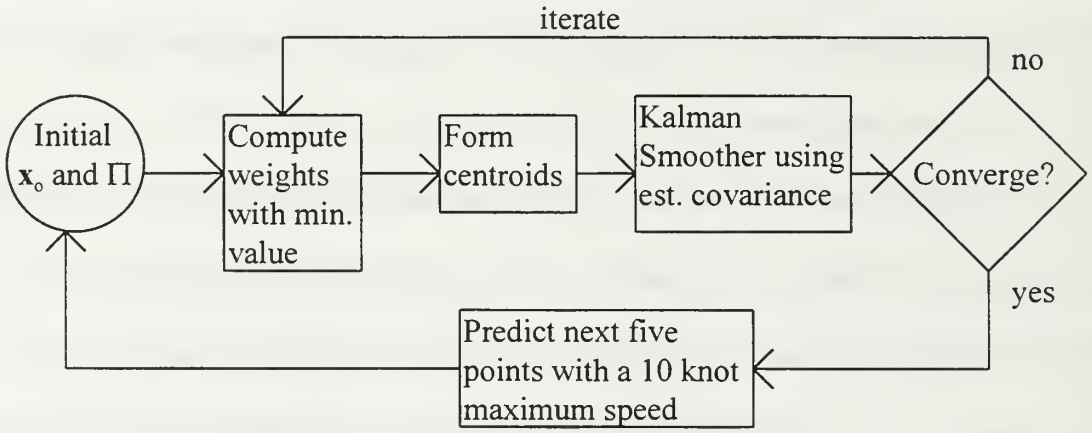

Figure 3.2. Modified PMHT

C. FURTHER MODIFICATIONS

The above modifications were sufficient in order for the PMHT algorithm to work well. With these modifications, the PMHT algorithm was able to perform better than the PDAF at clutter densities up to 3.33×10^{-2} clutter points per square kilometer. This value for the clutter density appears to be low. However, the standard deviation for the range is +/-100 meters, and the standard deviation for the bearing is +/-3 degrees. Therefore, with these values in the measurement covariance matrix, this clutter density is fairly significant.

On the other hand, in order to increase the performance of the algorithm further, the following modifications were also implemented and found to be effective, especially for the higher clutter densities.

1. Clutter Weight Model Inflation

Adjusting the clutter weight model parameter r after the algorithm has sorted out the first 10 points has proven beneficial. That is, for the first 10 points, ρ is set to a value of 10^{-12}, and then it is increased to a value of 10^{-10} for the rest of the batches.

2. Extended Kalman

The numerical complications involved with the measurement covariance matrix led to trying the Extended Kalman algorithm. As has been discussed, the classical Kalman requires a different covariance matrix for each different point in the Cartesian Coordinates. On the other hand, if the Extended Kalman is used, one measurement covariance matrix is used for all models at all time scans. Using this approach has shown some significant advantages which will be discussed in greater detail in Chapter V.

Therefore, the changes necessary to implement the Extended Kalman smoother are the following:

a. Coordinate Conversion

First, the innovations equation (Eqn. 3.3) must incorporate the coordinate conversion.

$$\tilde{\mathbf{z}}_{mlr}^{(i)} = \mathbf{z}_{lr} - \mathrm{g}(\mathbf{x}_{lm}^{(i)}) \tag{3.22}$$

where,

$$g(\mathbf{x}) = \begin{bmatrix} \sqrt{x^2 + y^2} \\ \text{atan}\left(\frac{y}{x}\right) \end{bmatrix} \tag{3.23}$$

b. The Measurement Matrix

The Kalman measurement matrix (Eqn. 3.5) will no longer be a constant matrix, but will have to be re-computed each time.

$$\mathbf{C}_{lm}^{(i)} = \left[\frac{\partial g}{\partial x}(\mathbf{x}_{lm}^{(i)})\right] = \begin{bmatrix} \frac{x}{r} & 0 & \frac{y}{r} & 0 \\ \frac{-y}{r^2} & 0 & \frac{x}{r^2} & 0 \end{bmatrix} \tag{3.24}$$

The same modifications are also necessary for the smoothing step. Therefore, (Eqns. 3.11-3.16) will be changed accordingly. Another effect of using the Extended Kalman is that the algorithm converges more quickly and easily than before. Therefore, the stopping criteria, *Kc*, can be lowered, producing relatively the same number of iterations, while allowing a tighter convergence parameter. For the Extended Kalman algorithm, *Kc* was set to 10^{-8}.

3. Stricter Speed Limit

As different geometries were simulated, I found that the original speed limit still allowed targets to "run away," i.e., have a velocity estimate which was too fast. Therefore, a modification to the limit on the velocity of the predicted track estimate was necessary.

The stricter speed limit is implemented by first selecting the middle time scan estimate as the baseline. If this estimate has a speed in excess of 10 knots, then the velocity of the baseline is reduced to reflect a speed of 10 knots. Then the state estimates at both past and future times are generated from the adjusted baseline state with its new

refined velocity. These estimates are then used in the next batch of processing. Even with this stricter speed limit, the algorithm is still allowed to converge to a final track estimate with any velocity.

4. Attribute Data

The use of measurement attribute data was explored and added to the algorithm. This method assumes that a measure of target amplitude was available in addition to the range-bearing measurement. The measured amplitude is drawn from a Rayleigh distribution with specified mean, which has the form:

$$f(a) = \left(a / \sigma^2\right)\exp\left(-a^2 / 2\sigma^2\right),\ a > 0 \qquad (3.25)$$

The mean for this distribution is $a\sqrt{\pi/2}$. For this research, targets and clutter are assigned different means. The clutter model is given a mean with a=1, and the target models have means with a>1. This data is included by multiplying the weights found in (Eqns. 3.1 & 3.17) by the Rayleigh distribution. Therefore, (Eqns. 3.1 & 3.17) become

$$w_{mlr}^{*(i+1)} = f(a)\frac{\exp\left(-\frac{1}{2}\tilde{\mathbf{z}}_{mlr}^{(i)\mathrm{T}}\Sigma_{lmi}^{-1}\tilde{\mathbf{z}}_{mlr}^{(i)}\right)}{2\pi\sqrt{\det(\Sigma_{lmi})}} \qquad (3.26)$$

$$w_{mlr}^{*(i+1)} = f(a)\rho \qquad (3.27)$$

respectively. This modification was only used in the simulations that are indicated in Chapter V.

A block diagram with all of the modifications, except the attribute data, is shown in Figure 3.3.

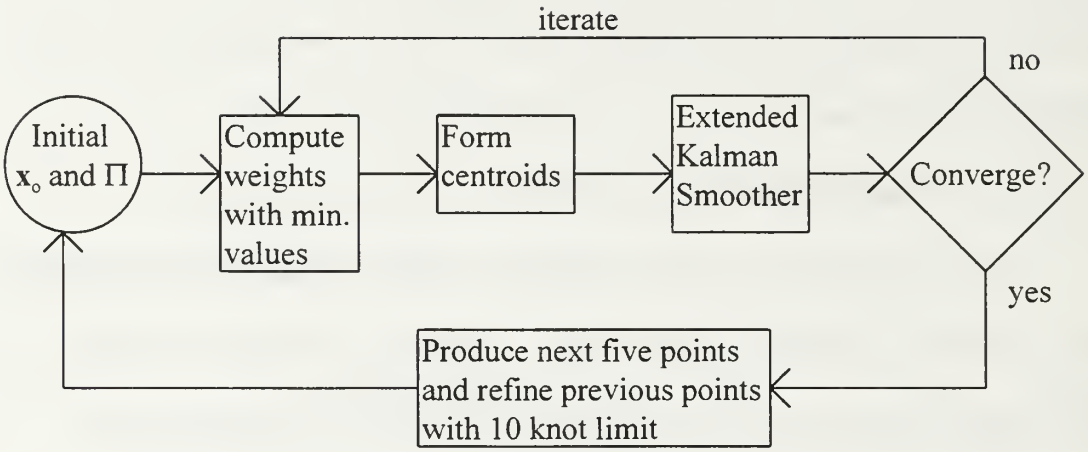

Figure 3.3. Improved PMHT

This concludes the layout of the PMHT algorithm, which is utilized in this research. In the next chapter, I will discuss the initialization routine that is used for this algorithm.

IV. TARGET STATE INITIALIZATION

A. GENERAL DISCUSSION

The initialization algorithm used in this research is of the *N*-of-*N* variety. That is, for the first *N* time scans, there must be *N* measurement points which meet the gating criteria. The PMHT algorithm is very sensitive to the initial estimates it is given. For small values of *N*, the initial heading can be extremely inaccurate, and then the algorithm is less likely to converge to the true track. Therefore, I have found that values of $N \geq 5$ are necessary for the PMHT algorithm to perform reliably.

B. THE N-of-N INITIALIZATION ALGORITHM

There are two steps for this process when $N \geq 3$. The first step is a gating equation which eliminates all points which are not likely to be associated as a track. Then once a set of *N* points has been designated as a new track, a least squares algorithm is used to fit the optimal track to these *N* points.

1. Gating

As has been stated before, this research assumes that targets have a speed between 2 and 10 knots. If the target velocity vector is known and two successive measurements ($\mathbf{z}_t$ and $\mathbf{z}_{t+1}$) are obtained for the target, then the quantity

$$\chi^2 = (\mathbf{z}_{t+1} - \mathbf{z}_t)^{\mathrm{T}}[\mathbf{R}_{t+1} + \mathbf{R}_t]^{-1}(\mathbf{z}_{t+1} - \mathbf{z}_t) \tag{4.1}$$

has a non-central Chi-squared distribution, where $\mathbf{z}_t$ and $\mathbf{R}_t$ are the measured track position and measurement covariance matrix at time t. It was empirically demonstrated that for the target speeds, tracking ranges, and measurement errors used in this research, a cut-off of $\chi^2 \leq 50$ kept all target associations. This cut-off produces an effective gate for 2-of-2 association of approximately 75 square kilometers. A 3-of-3 case relies on a match (using the 2-of-2 algorithm) for the first and second measurements and the second and third measurements. Then a cut-off value for track formation was applied with $\chi^2 \leq 20$. This gives an association probability for measurements derived from a real track of 0.9972. Higher order associations project the subsequent result into the future to determine likely measurements for the new association. At each projection for the next higher association, a match is performed and then a χ^2 rejection.

2. Least Squares Fit

This initialization algorithm uses a least squares fit assuming a constant velocity target during the initialization period. Therefore, for the 3-of-3 case, the first three measurements are given by:

$$\begin{aligned} \mathbf{z}_{1m} &= \mathbf{C}\mathbf{x}_{1m} + \mathbf{e}_{1r} \\ \mathbf{z}_{2m} &= \mathbf{C}\mathbf{x}_{2m} + \mathbf{e}_{2r} = \mathbf{C}[\Phi\mathbf{x}_{1m} + \mathbf{w}_{1m}] + \mathbf{e}_{2r} \\ \mathbf{z}_{3m} &= \mathbf{C}\mathbf{x}_{3m} + \mathbf{e}_{3r} = \mathbf{C}[\Phi(\Phi\mathbf{x}_{1m} + \mathbf{w}_{1m}) + \mathbf{w}_{2m}] + \mathbf{e}_{3r} \end{aligned} \tag{4.2}$$

Since a straight, constant velocity target is assumed during the initialization, the process errors are set to zero (i.e., $\mathbf{w}_{tm}$=0 for every t). Therefore,

$$\begin{bmatrix} \mathbf{z}_{1m} \\ \mathbf{z}_{2m} \\ \mathbf{z}_{3m} \end{bmatrix}_{6x1} = \mathbf{F}\mathbf{x}_{1m} + \begin{bmatrix} \mathbf{e}_{1r} \\ \mathbf{e}_{2r} \\ \mathbf{e}_{3r} \end{bmatrix} \tag{4.3}$$

where,

$$\mathbf{F} = \begin{bmatrix} \mathbf{H} \\ \mathbf{H}\Phi \\ \mathbf{H}\Phi\Phi \end{bmatrix} \tag{4.4}$$

where the covariance associated with the error vector is

$$\mathrm{E}\left\{ \begin{bmatrix} \mathbf{e}_{1r} \\ \mathbf{e}_{2r} \\ \mathbf{e}_{3r} \end{bmatrix} \begin{bmatrix} \mathbf{e}_{1r} & \mathbf{e}_{2r} & \mathbf{e}_{3r} \end{bmatrix} \right\} = \begin{bmatrix} \mathbf{R}_{1r} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{2r} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}_{3r} \end{bmatrix} \equiv \Sigma_{\mathbf{R}} \tag{4.5}$$

Therefore, the least squares estimate of $\mathbf{x}_{1m}$ is given by:

$$\hat{\mathbf{x}}_{1m} = \left[\mathbf{F}^{\mathrm{T}}\Sigma_{\mathbf{R}}^{-1}\mathbf{F}\right]^{-1}\mathbf{F}^{\mathrm{T}}\Sigma_{\mathbf{R}}^{-1}\begin{bmatrix} \mathbf{z}_{1m} \\ \mathbf{z}_{2m} \\ \mathbf{z}_{3m} \end{bmatrix} \tag{4.6}$$

and

$$\hat{\mathbf{x}}_{3m} = \Phi\hat{\mathbf{x}}_{2m} = \Phi\Phi\hat{\mathbf{x}}_{1m} \tag{4.7}$$

since this gives the estimates for the position. For the associated state estimate covariance matrices, the following equations are used:

$$\mathbf{P}_{1m} = \left[\mathbf{F}^{\mathrm{T}}\Sigma_{\mathbf{R}}^{-1}\mathbf{F}\right]^{-1}\mathbf{F}^{\mathrm{T}}\left(\left[\mathbf{F}^{\mathrm{T}}\Sigma_{\mathbf{R}}^{-1}\mathbf{F}\right]^{-1}\mathbf{F}^{\mathrm{T}}\Sigma_{\mathbf{R}}^{-1}\right)^{\mathrm{T}} \tag{4.8}$$

and

$$\mathbf{P}_{3m} = \Phi\mathbf{P}_{2m}\Phi^{\mathrm{T}} + \mathbf{Q} = \Phi\left(\Phi\mathbf{P}_{1m}\Phi^{\mathrm{T}} + \mathbf{Q}\right)\Phi^{\mathrm{T}} + \mathbf{Q} \tag{4.9}$$

From this 3-of-3 case, it is straightforward to expand this in order to apply this process to the 5-of-5 case. These estimates and their associated covariance matrices are then used in the first batch of the PMHT algorithm.

V. RESULTS AND COMPARISONS

A. SIMULATIONS

In this research, all of the comparisons are based on simulated data. Simulations were run for 30 time scans with the time between scans equal to 4 minutes. The 30 range-bearing target measurements are generated using additive Gaussian noise, where the range standard deviation is 100 meters and the bearing standard deviation is 3 degrees. An example of a simulated target run is shown in Figure 5.1.

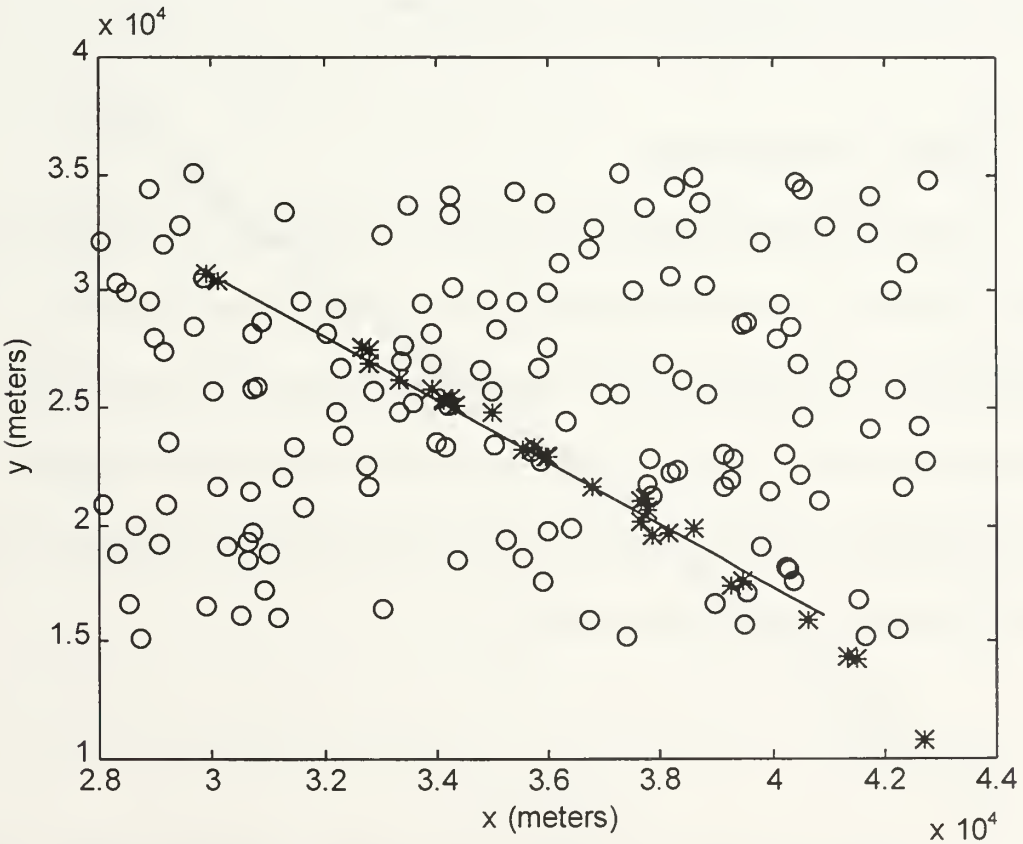

Figure 5.1. Straight Track with Clutter

The solid line is the true track, and the "*" are the noisy measurements. The actual target velocity is 5 knots in all scenarios. The sensor is located at (0,0), so the target is moving predominantly in the cross range dimension at a range greater than 40 kilometers.

This example also depicts clutter points as circles. Clutter was generated in a uniform random fashion throughout the area of interest. For this example the clutter density is 1.67×10^{-2} clutter points per square kilometer, or in other words, there are five clutter points for each time scan. Clutter densities in this study were varied from 3.33×10^{-3} to 6.67×10^{-2} clutter points per square kilometer.

B. OTHER ALGORITHMS

The MHT and PDAF algorithms were used as a benchmark to see how well the PMHT is performing. As was mentioned in Chapter I, both of these are established algorithms that are currently being used in tracking applications. The results from these algorithms were produced using the exact same scenarios as were used for the PMHT. The MHT and PDAF algorithms used are widely discussed in several different texts (e.g., Bar-Shalom and Li [Ref. 4], Blackman [Ref. 5]).

C. RESULTS

Three different target motion geometries were tested during this research. First, there is the basic straight line track as is depicted in Figure 5.1. Second, two crossing

tracks are used moving in straight lines with constant velocity. The third geometry is a track which makes a turn half way through the simulation.

The PMHT algorithm was first tested in each of the three geometries with a low clutter density (3.33×10^{-3} clutter points per square kilometer). Then it was confirmed with higher clutter densities and compared to the competing algorithms.

1. Straight Line Target Motion

Figure 5.2 displays a typical converged result produced by the PMHT algorithm.

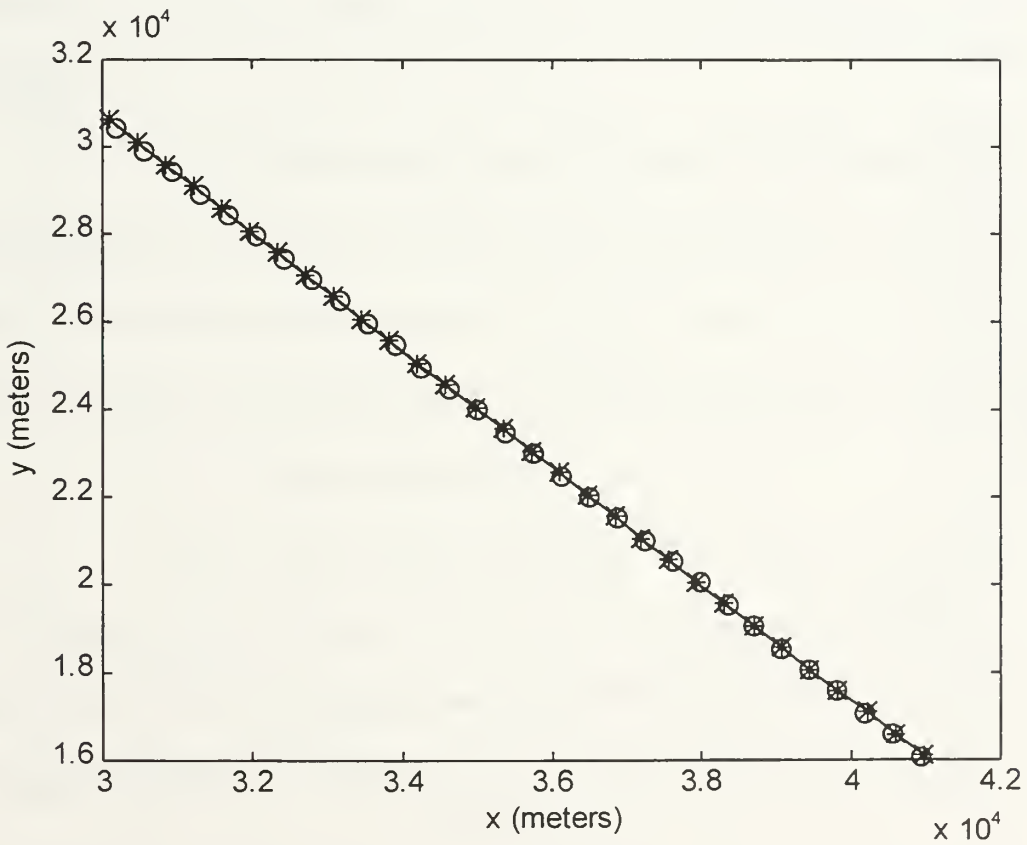

Figure 5.2. Typical PMHT Result

On this single, straight line track, the circles are the actual target positions at each time increment, and the "*" symbols are the smoothed, converged estimates. The clutter density for this simulation is 3.33×10^{-3} clutter points per square kilometer.

a. Low Clutter Density

In order to quantify the results, mean distance errors were computed for the final target estimates at each time scan. The means were taken from 500 simulation runs. Figure 5.3 shows these mean distance errors for a straight line track in a clutter density of 3.33×10^{-3} clutter points per square kilometer.

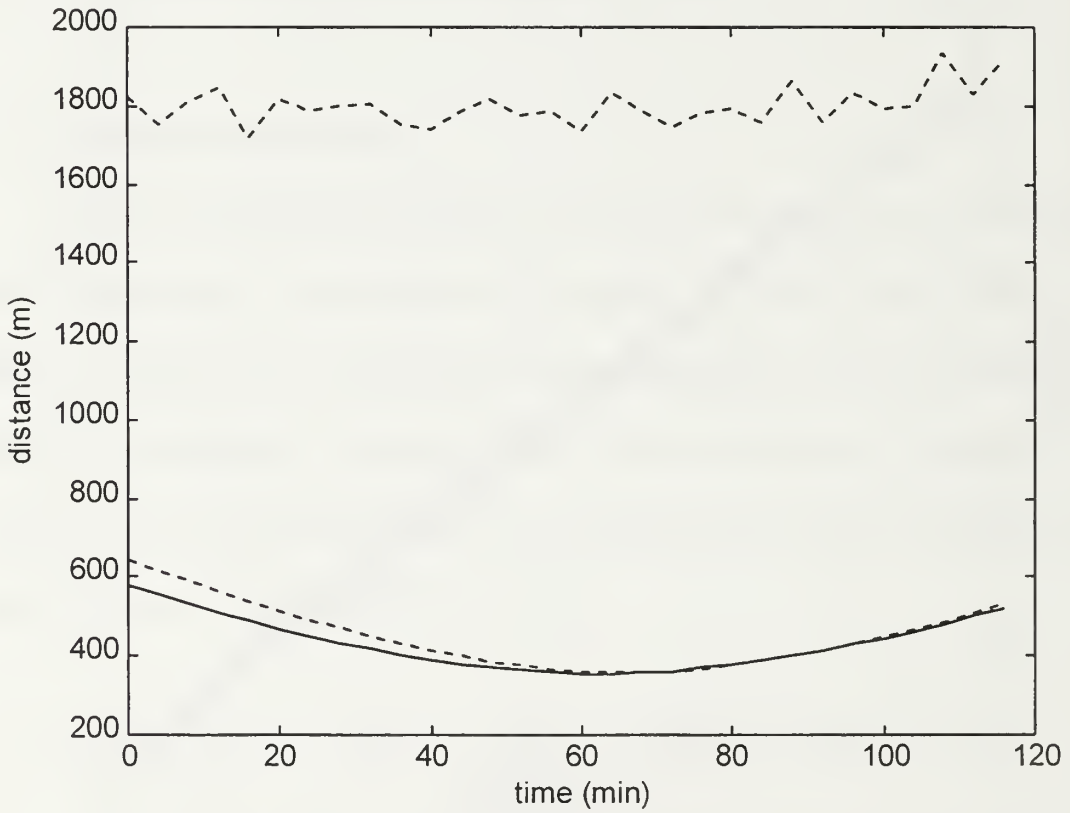

Figure 5.3. Measurement and Estimate Errors (low clutter)

The lowest dotted line curve on the figure is the result of running the Kalman Smoother using the actual target measurements (i.e., no clutter) and computing the mean distance errors over 1000 simulation runs. Therefore, this curve represents an approximate theoretical minimum for algorithm performance in the absence of clutter. The highest curve on the figure (the dotted line) is the average, noisy measurement error computed over 1000 runs. Clearly, an algorithm should be below this line to be considered as a viable option. The other line on the figure (the solid line) is the mean estimate error, which was produced by the algorithm. This is the average estimate error for 500 simulation runs at a clutter density of 3.33×10^{-3} clutter points per square kilometer.

In this simulation, the algorithm happened to produce estimate errors which have a better mean than the Kalman Smoother without clutter. This can be true because of randomness, but in general will not happen. However, it does show that the PMHT algorithm is performing extremely well in low clutter, as would be expected.

b. Medium Clutter Density

Figure 5.4 shows the results using the PMHT algorithm in a medium clutter density. The medium clutter density has five clutter points and a noisy measurement in the search area at each time scan. The clutter density is 1.67×10^{-2} clutter points per square kilometer. The solid line in Figure 5.4 displays the results from using the Extended Kalman Smoother (EKS) in the algorithm. The broken line shows the results of using the conventional Kalman Smoother in Cartesian Coordinates, using the debiased equations. As can be seen, the Extended Kalman Smoother performed better.

Furthermore, the EKS performed the 500 simulations in only 204 minutes, whereas, the conventional Kalman algorithm took 280 minutes. Therefore, from these results, I have determined for this application that it is better to use the EKS as opposed to the conventional Kalman with the debiased equations. From here on, I will only show results from the EKS, but at other clutter densities, the results are similar between the conventional Kalman and the EKS, as is shown here.

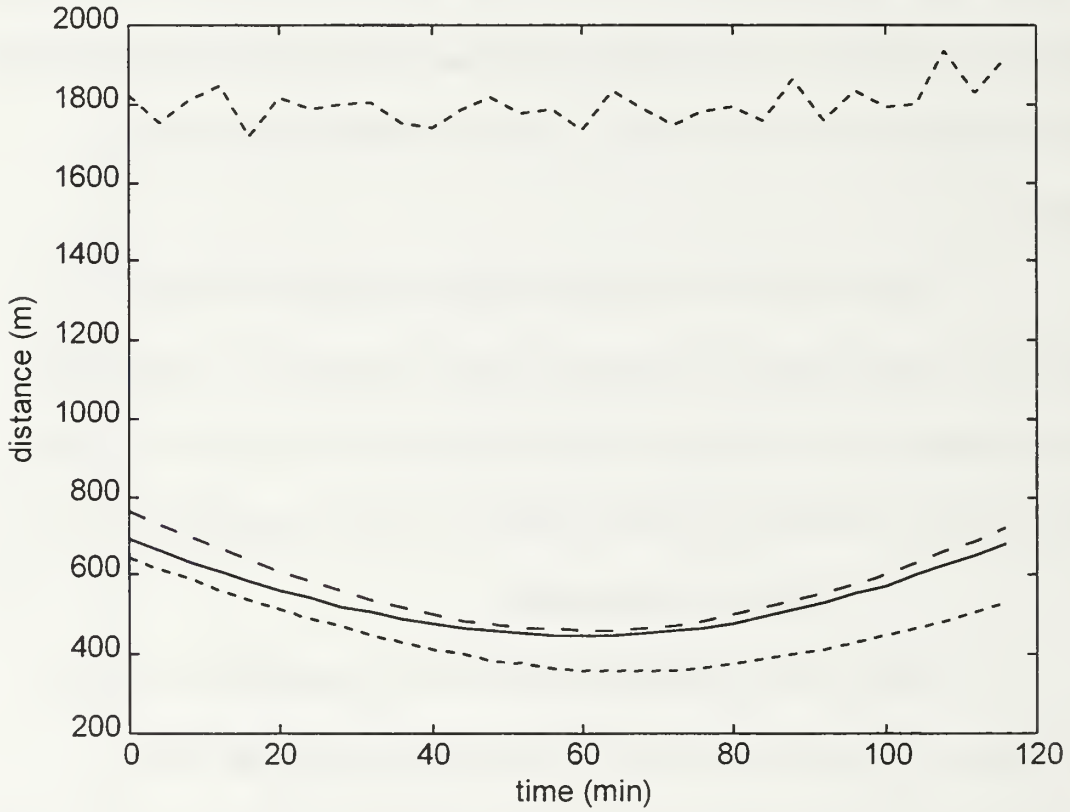

Figure 5.4. Comparison between EKS and Conventional Kalman

Figure 5.5 shows the comparison between the nearest neighbor (NN), PMHT, MHT, and PDAF. On this figure, there are four different lines—one for each of

the different algorithms. The solid line represents the PMHT; the solid line with circles is the nearest neighbor; the broken line is the MHT; and the dash-dot line is the PDAF. The important estimate error to compare is at the last time scan. This is the time that is current and is of importance. The PMHT will almost always have better errors for earlier time scans because of the smoothing process. For this clutter density, the PMHT is performing slightly better than the MHT and considerably better than the PDAF and the nearest neighbor. This is especially promising given the fact that the PMHT is less complicated and easier to compute than the MHT. Since the nearest neighbor algorithm is clearly not a viable option, it is not included on subsequent plots.

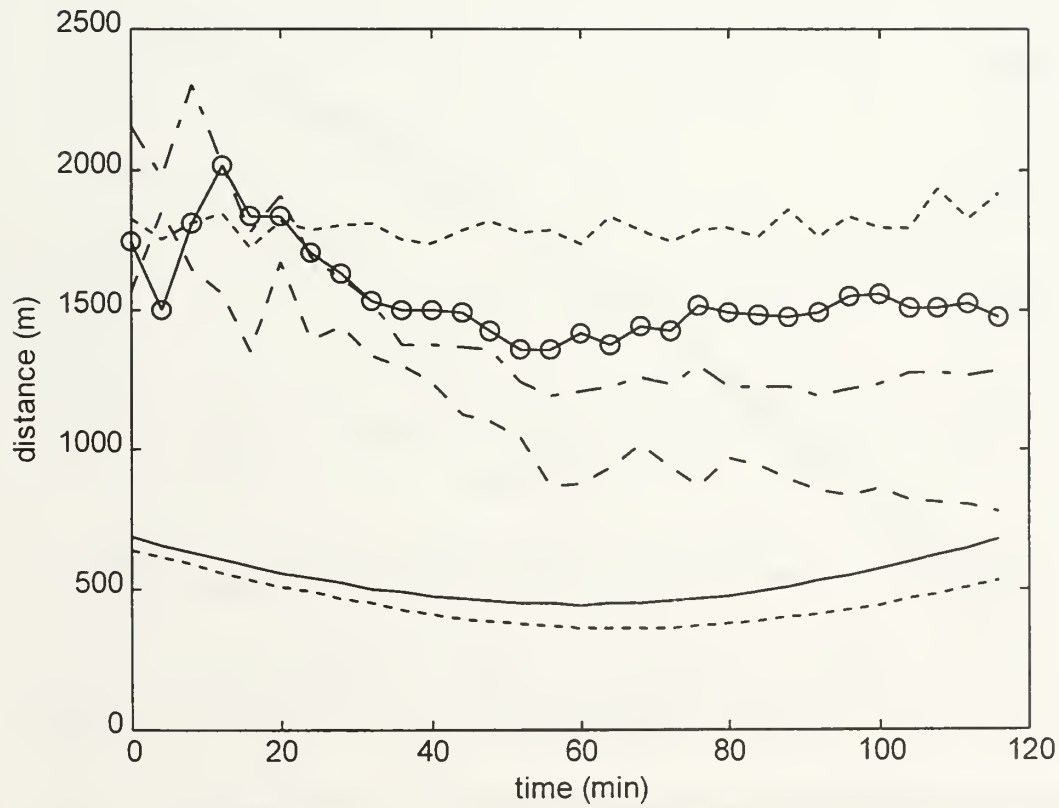

Figure 5.5. Comparison between NN, PMHT, PDAF, and MHT (medium clutter)

c. High Clutter Density

For a high density clutter environment, a clutter density of 3.33×10^{-2} clutter points per square kilometer was used. This meant that there were 10 clutter points and one noisy measurement in the search area per time scan. The results from this simulation are shown in Figure 5.6.

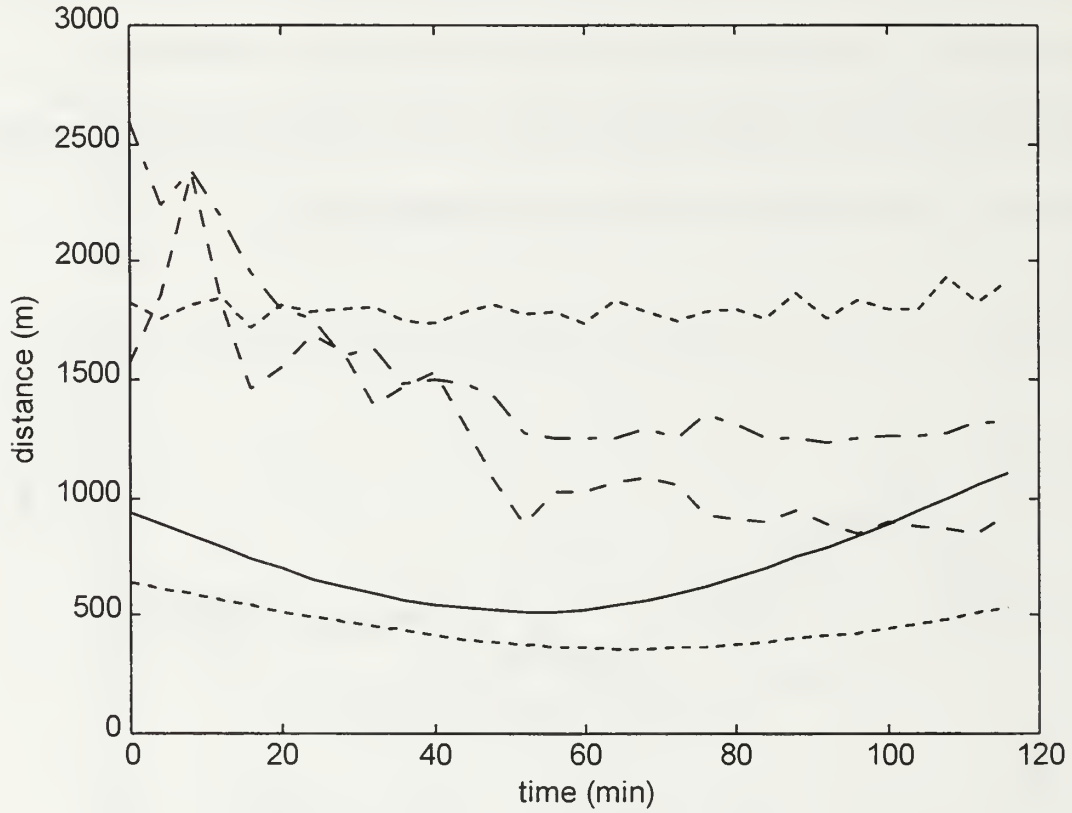

Figure 5.6. Comparison between PMHT, PDAF, and MHT (high clutter)

These results show that the PMHT is performing better than the PDAF, but worse than the MHT for this clutter density. Therefore at this point, there is a cost trade-off versus the algorithm performance. The MHT performs the best, but it also is the

most expensive in terms of computations. On the other hand, the PDAF is producing the worst results, but it is the cheapest and fastest to calculate.

2. Turning Tracks

For this scenario, the target moved in a straight line with constant speed for the first 60 minutes. At that point, the target made a 10 degree turn and then continued in a straight line for the remaining time. This target motion and a typical converged result produced by the PMHT algorithm are displayed in Figure 5.7.

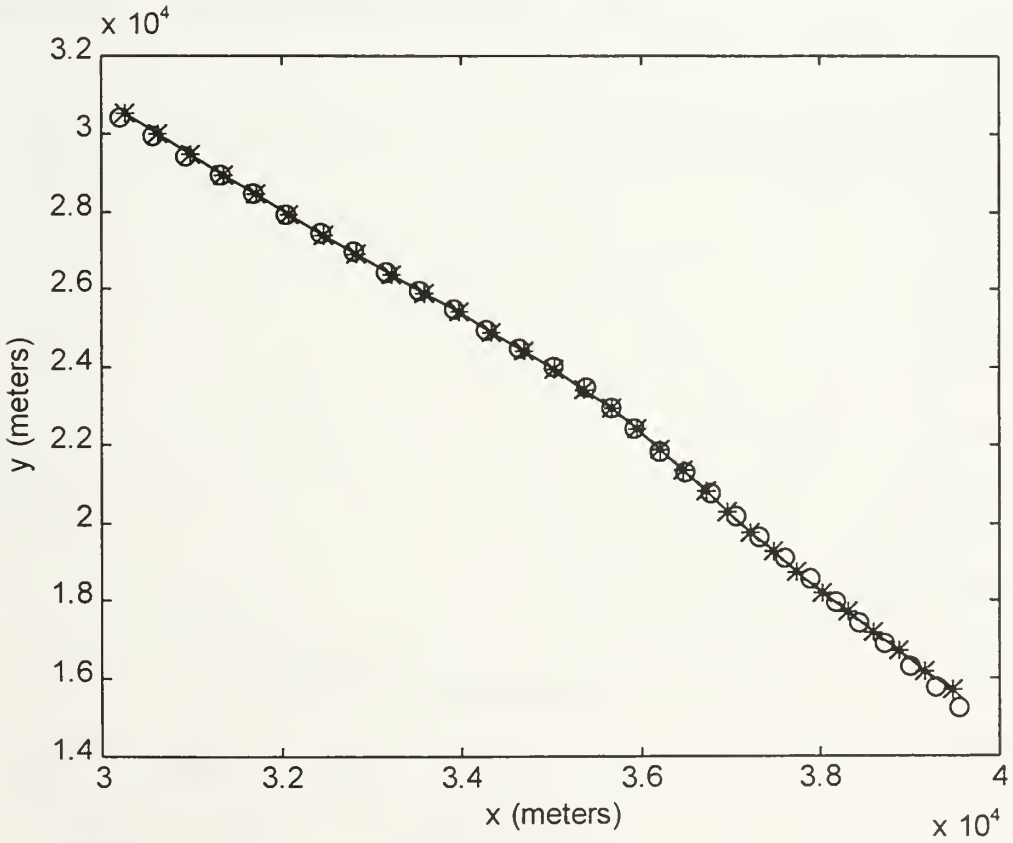

Figure 5.7. Typical PMHT Results with a Turning Track

Again, the circles are the actual target positions at each time increment, and the "*" symbols are the smoothed, converged estimates. The clutter density for this simulation is 3.33×10^{-3} clutter points per square kilometer. This result shows that the algorithm is not handling the turn quite as well as the straight line track; however, it is tracking nonetheless. For this 10 degree turn, the q factor was set to a value of one.

a. Low Clutter Density

Figure 5.8 displays the mean errors for a turning track for 500 simulation runs. The solid line represents the estimate errors at each time scan.

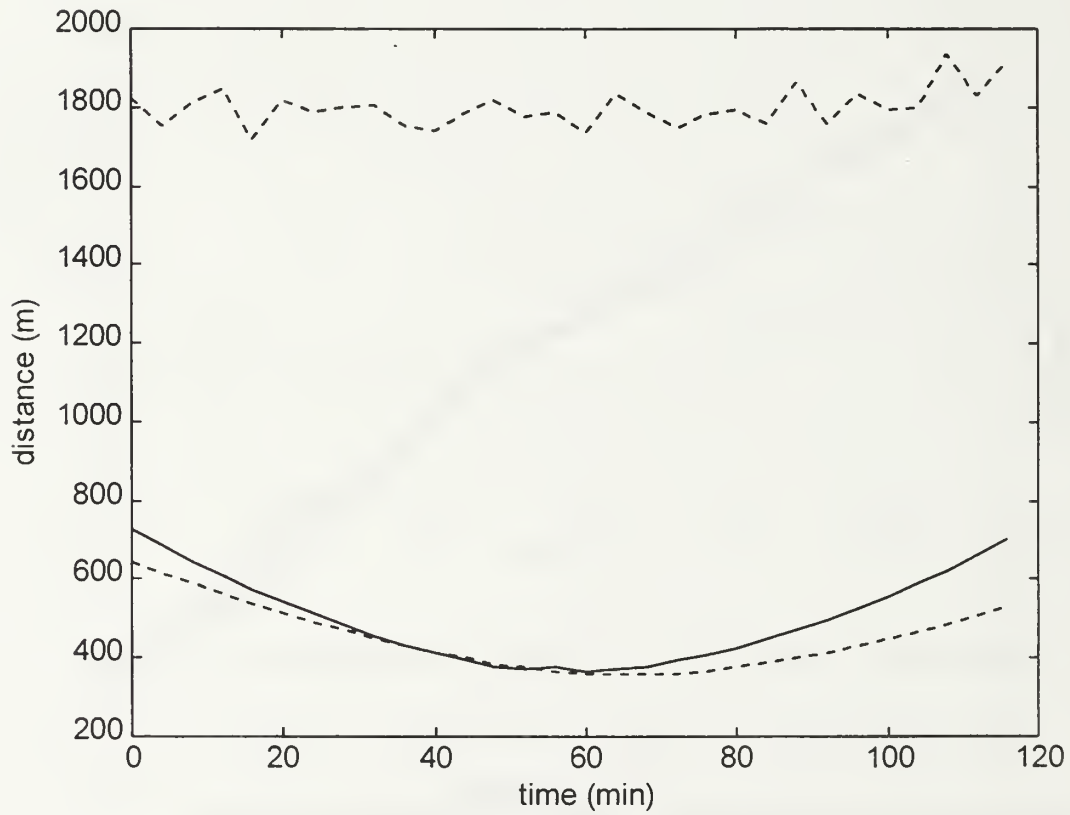

Figure 5.8. Measurement and Estimate Errors (low clutter)

The clutter density for this simulation is 3.33×10^{-3} clutter points per square kilometer. While the errors for this scenario are higher than for the straight track, they are still well below the measurement errors.

b. Medium Clutter Density

Figure 5.9 shows the results of the PMHT algorithm on a turning track in a clutter density of 1.67×10^{-2} clutter points per square kilometer. These results show that the algorithm is still performing well even with the target making a turn in the presence of a medium clutter density. This further justifies the use of the q factor as a small, but positive value. For this scenario, q was set to a value of 10.

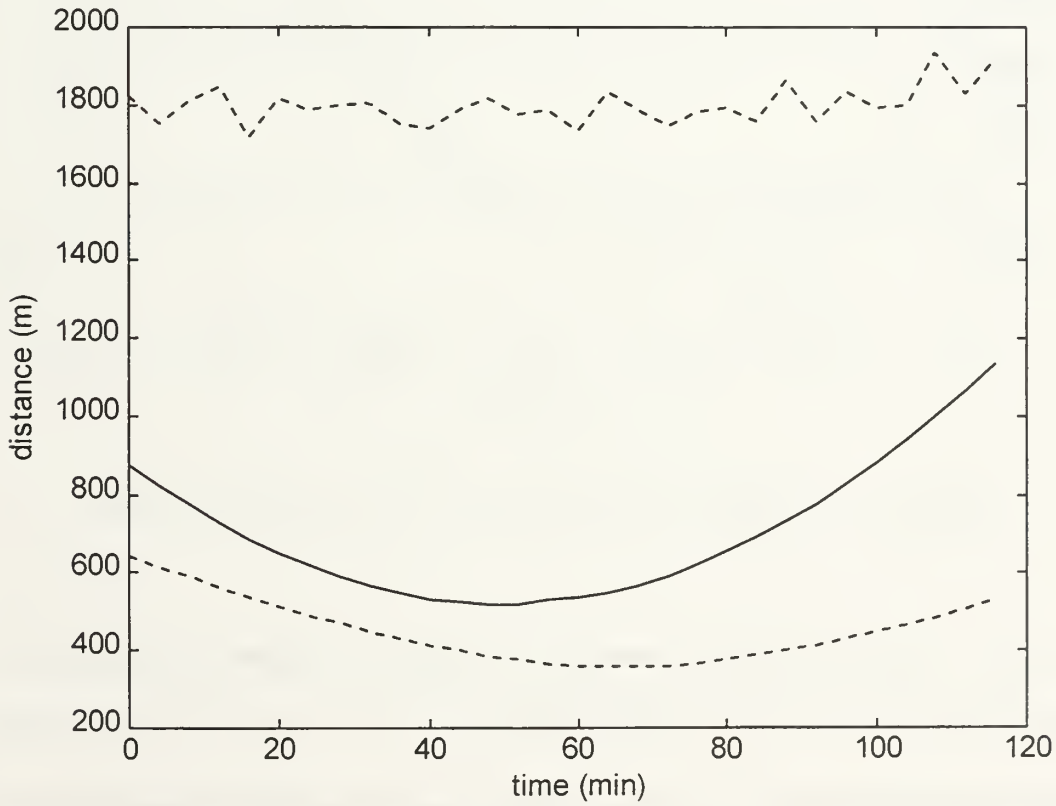

Figure 5.9. Measurement and Estimate Errors (medium clutter, 10° turn)

Figure 5.10 shows the results of the three algorithms estimating a turning track with a 90 degree turn instead of a 10 degree turn. Again, the solid line represents the PMHT; the broken line is the MHT; and the dash-dot line is the PDAF. The clutter density is still 1.67×10^{-2} clutter points per square kilometer.

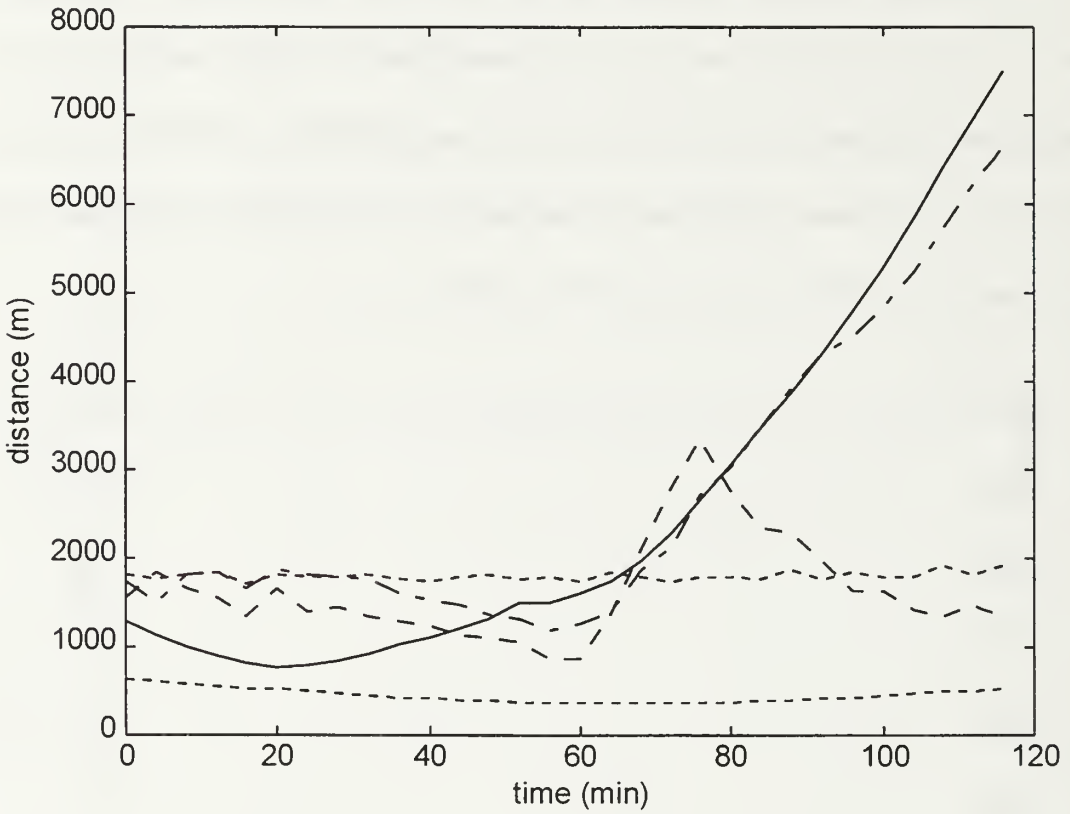

Figure 5.10. Comparison between PMHT, PDAF, and MHT (90° turn)

Here the q factor was set to a value of 300 in an effort to try and get the PMHT algorithm to track through the turn. However, with a turn this radical, the PMHT and PDAF algorithms were unable to maintain the target. The MHT results show that the target was lost at the turn, but was able to be reacquired after the turn. The MHT

algorithm has reacquisition logic inherent in the algorithm itself, while the other two algorithms do not have this process inherent to the basic tracking filter. Therefore, this result leads to the conclusion that the use of the PMHT will require that some sort of target re-initialization be implemented in a fielded system. A procedure for linking the new track with the old would also be required.

c. High Clutter Density

Figure 5.11 shows the results of the PMHT tracking a target through a 10 degree turn in a clutter density of 3.33×10^{-2} clutter points per square kilometer.

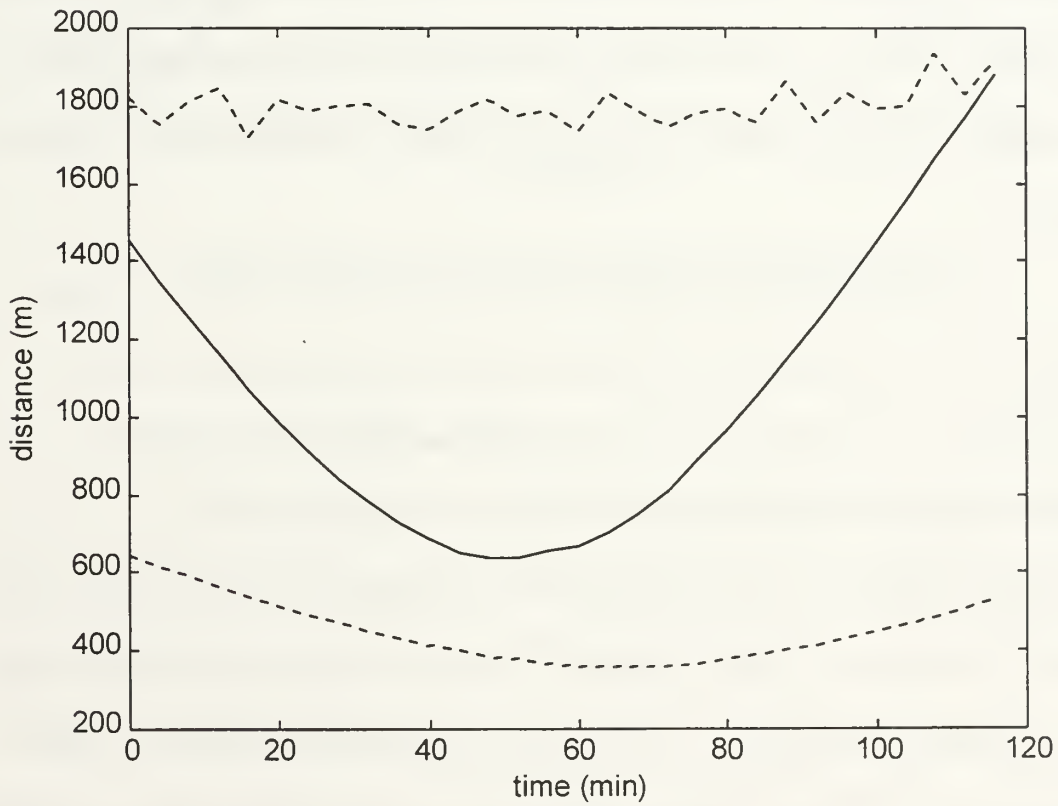

Figure 5.11. Measurement and Estimate Errors (high clutter, 10° turn)

These results confirm what was seen at the medium clutter density—namely that the PMHT algorithm is able to robustly follow a track as it makes small maneuvers. This justifies further using the small, but non-zero q factor value instead of setting it to zero, which would be the equivalent of using a least squares fit instead of a Kalman filter or smoother.

3. Crossing Tracks

In this scenario, there are two targets in the simulation. Target one executes the same track that was used in the straight track from before. Target two executes a track which starts in the bottom left-hand corner of the figure and runs toward the upper right-hand corner. Therefore, this new track is predominantly in the cross bearing direction. These tracks along with typical converged results from the PMHT algorithm are shown in Figure 5.12.

Again, the actual target positions are the circles, and the "*" symbols are the smoothed, converged estimates. The clutter density for this simulation is 3.33×10^{-3} clutter points per square kilometer. Initially, there were some problems with the algorithm tracking the different trajectories, but since the improved limit on velocity was implemented, this has not been a drawback. This typical result shows the effect of the greater uncertainty in the bearing dimension. Track one, which is predominantly moving in the cross-range direction, has a greater uncertainty along its axis of travel. On the other hand, track two, which is predominantly moving in the cross-bearing direction, has a greater uncertainty perpendicular to its axis of travel.

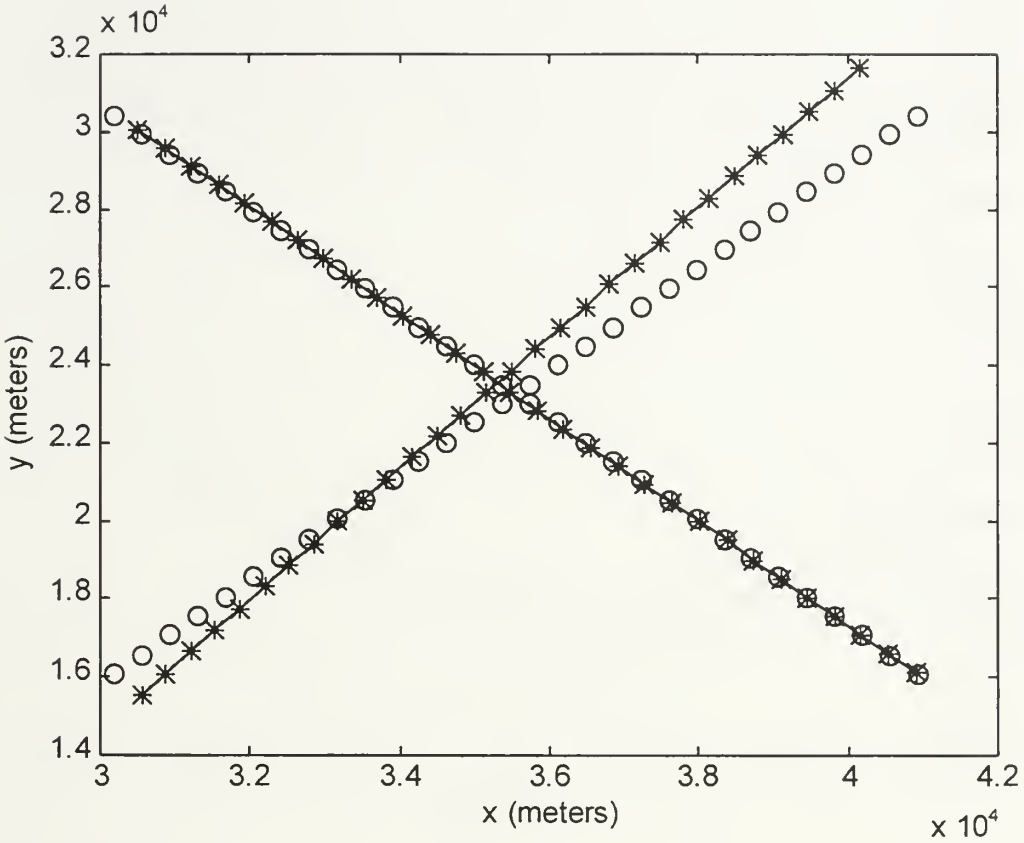

Figure 5.12. Typical PMHT Results with Crossing Tracks

a. Low Clutter Density

Figure 5.13 displays the results from the crossing track scenario with a low clutter density of 3.33×10^{-3} clutter points per square kilometer. The solid line shows the mean errors from track one, and the broken line is from track two. Again, the lowest dotted line represents the theoretical minimum, which was produced by running the Kalman smoother with just the noisy measurements. The highest line is the average measurement error over 1000 simulations. These results are still below the measurement error, but significantly higher than the single track in a low clutter density.

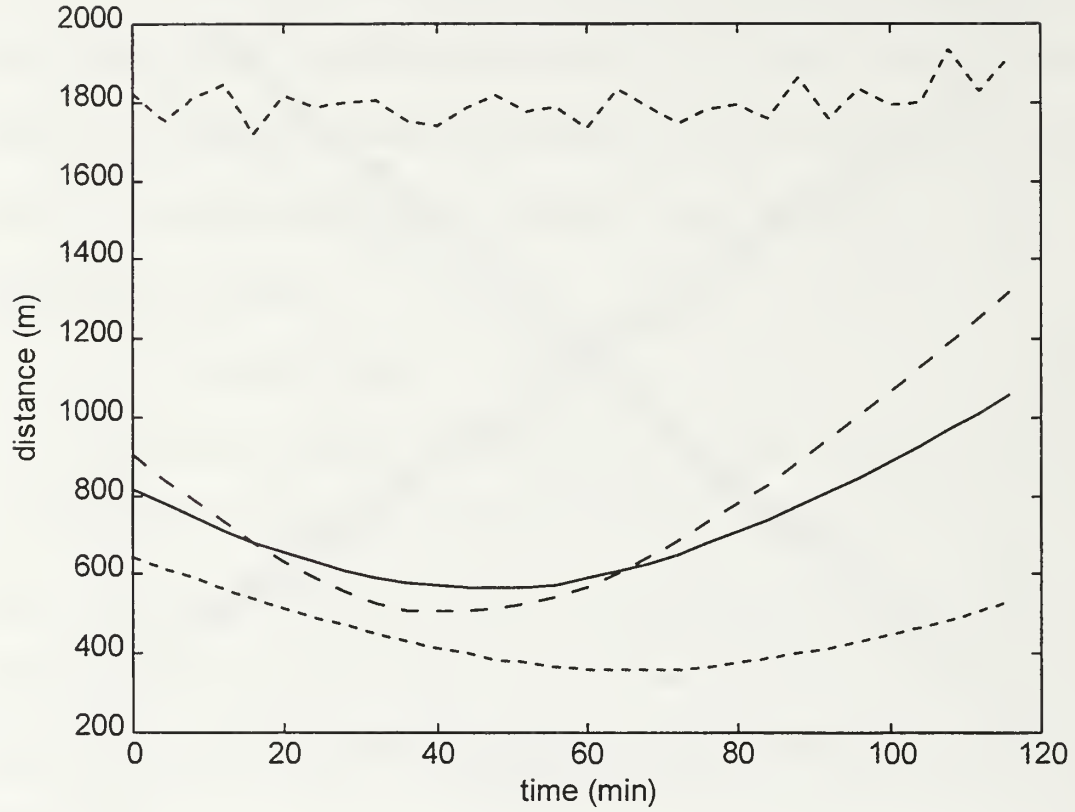

Figure 5.13. Errors for Crossing Tracks (low clutter)

One reason for this is the effect that each track has on the other one. The clutter has an equal probability of affecting the estimate to one side or the other, and, in general, clutter will tend to have little bias effect on the track estimate. On the other hand, the measurements from the other track will tend to pull the estimate toward these measurements, causing a bias to one side. Since, in this low clutter density the clutter has minimal effect, this bias effect of the other track is clearly evident. Figure 5.14 shows this bias effect. This figure is the average track estimate, which is produced by the algorithm over the 500 runs.

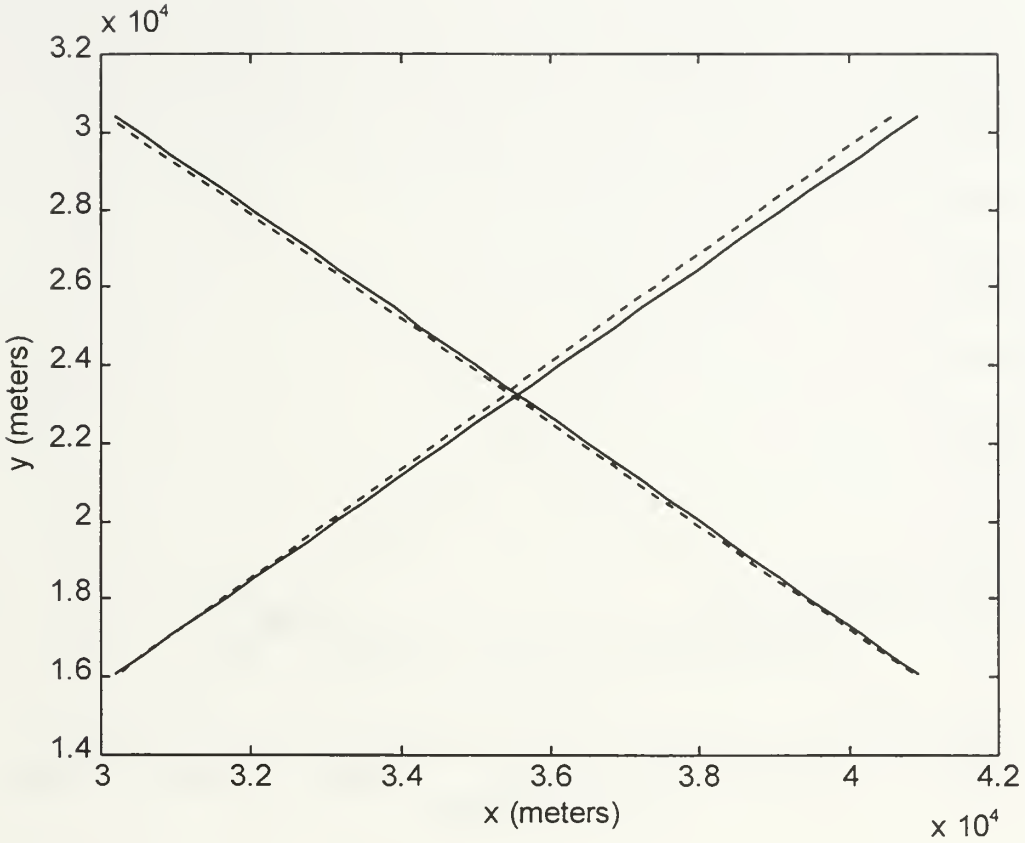

Figure 5.14. Average Track Estimate (low clutter)

The true target tracks are shown as the solid lines, and the estimate average is displayed as the dotted lines. Track one is pulled downward, and track two is pulled up towards track one. In the next subsection, I will show how more clutter will dampen out this bias effect.

b. Medium Clutter Density

Figure 5.15 shows the results from the crossing track scenario in a medium clutter density of 1.67×10^{-2} clutter points per square kilometer. Once again, the solid line represents the mean errors from track one, and the broken line is from track two.

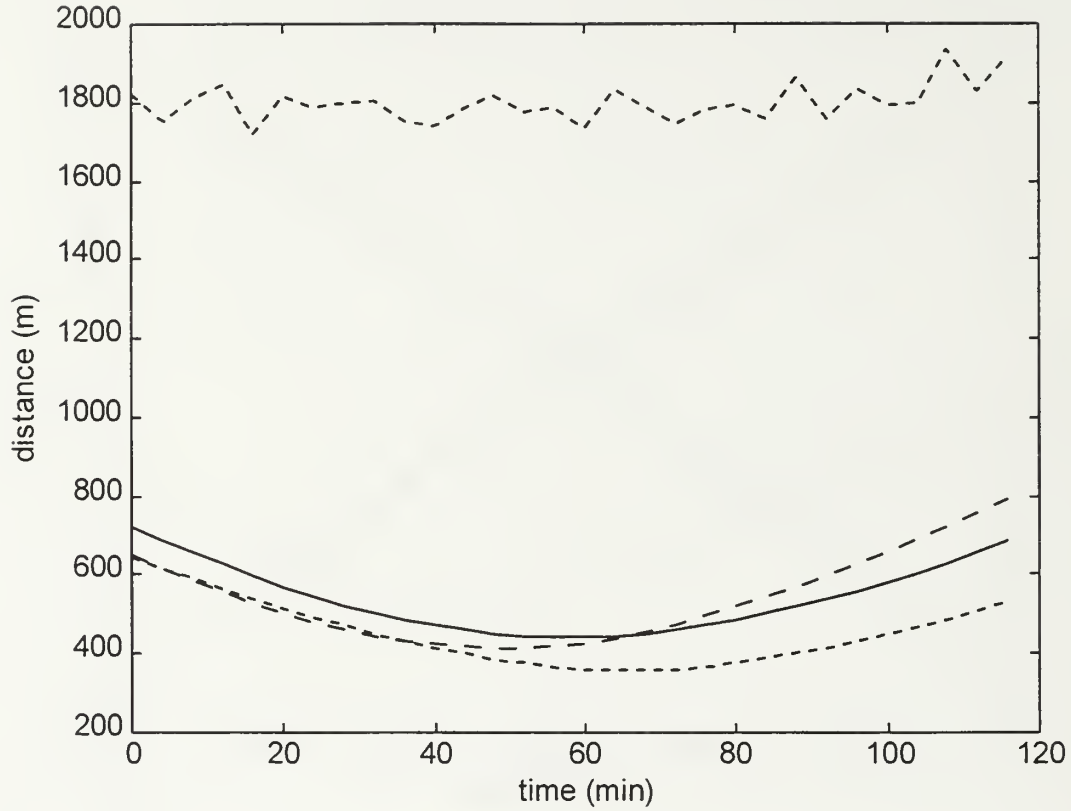

Figure 5.15. Errors for Crossing Tracks (medium clutter)

Even though the clutter density is five times greater, these results are significantly better than the low clutter density case. This is due to the higher clutter density dampening out the bias caused by the two tracks. These results compare very closely to those obtained for the single straight track in a medium clutter density.

Figure 5.16 shows the average track estimate for this scenario as was depicted in Figure 5.14 for the low clutter density. Again, the true target tracks are shown as the solid lines, and the estimate averages are displayed as the dotted lines. Indeed, the separate tracks show very little bias toward the other track in the simulation.

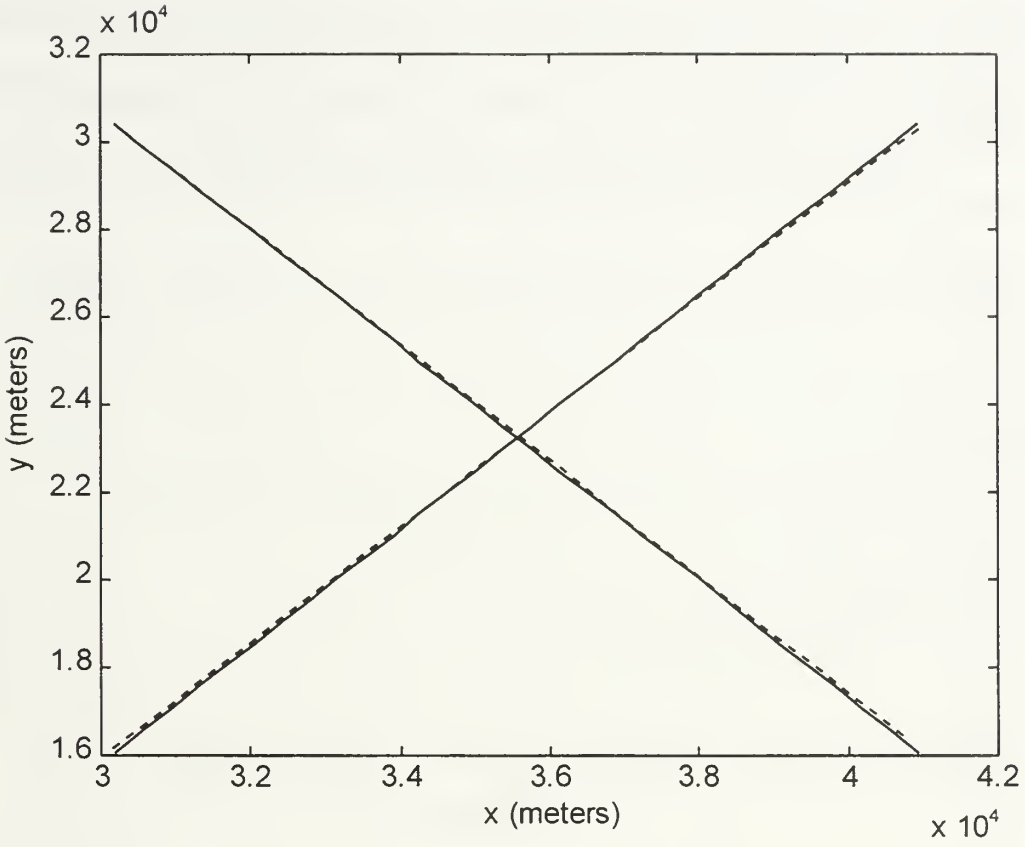

Figure 5.16. Average Track Estimate (medium clutter)

c. High Clutter Density

Figure 5.17 shows the results from the crossing track scenario in a high clutter density. Again, the solid line displays the mean error from track one, and the broken line represents the mean errors from track two. The clutter density for this scenario is 3.33×10^{-2} clutter points per square kilometer. For this simulation, the errors are starting to get up close to the measurement errors, and for clutter densities higher than this, the errors begin to exceed the measurement errors.

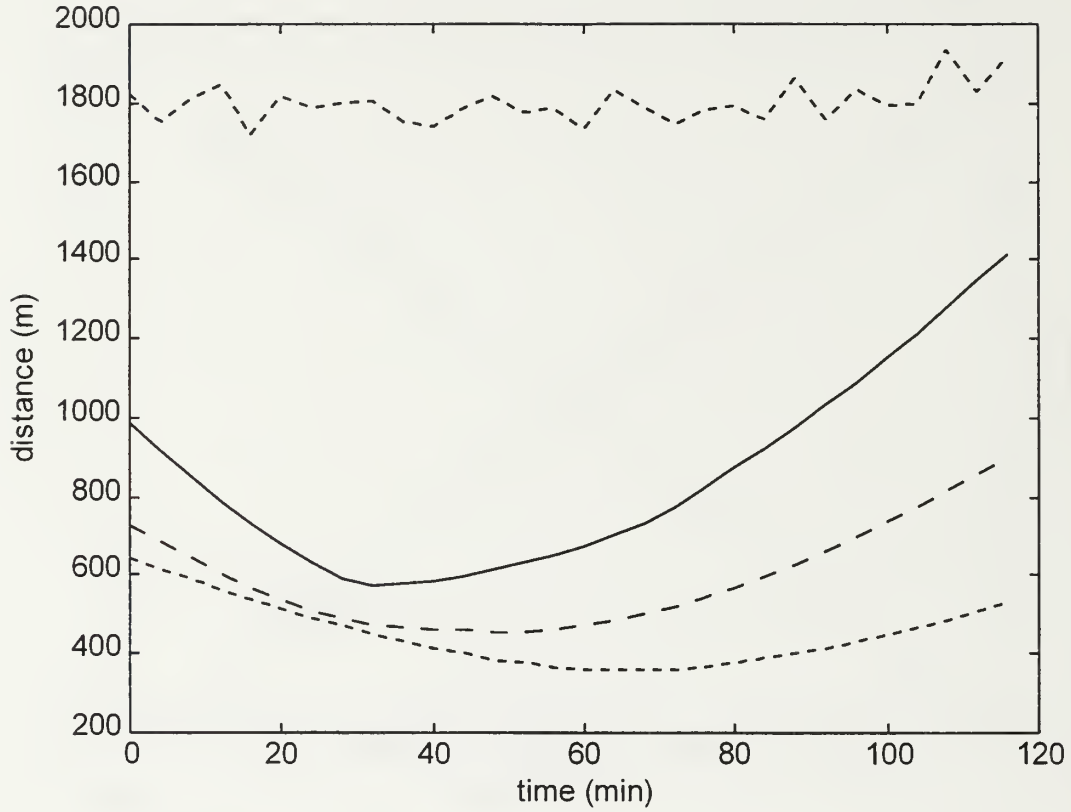

Figure 5.17. Errors for Crossing Tracks (high clutter)

Another observation from these results is the fact that track one's estimate is higher than that of track two. For the higher clutter densities, this was found to be the norm. The reason track one produces the higher estimate errors is due to the target motion being predominantly in the cross-range direction. Therefore, this is the reason track one was used for the majority of the simulations.

4. Attribute Data

The use of attribute data was researched to determine what level amplitude of target data, in relation to the clutter data amplitude, was necessary in order to improve the

performance of the PMHT algorithm. For this simulation, a very high clutter density of 6.67×10^{-2} clutter points per square kilometer was used. Figure 5.18 shows the results of the PMHT algorithm in this clutter density with and without attribute data.

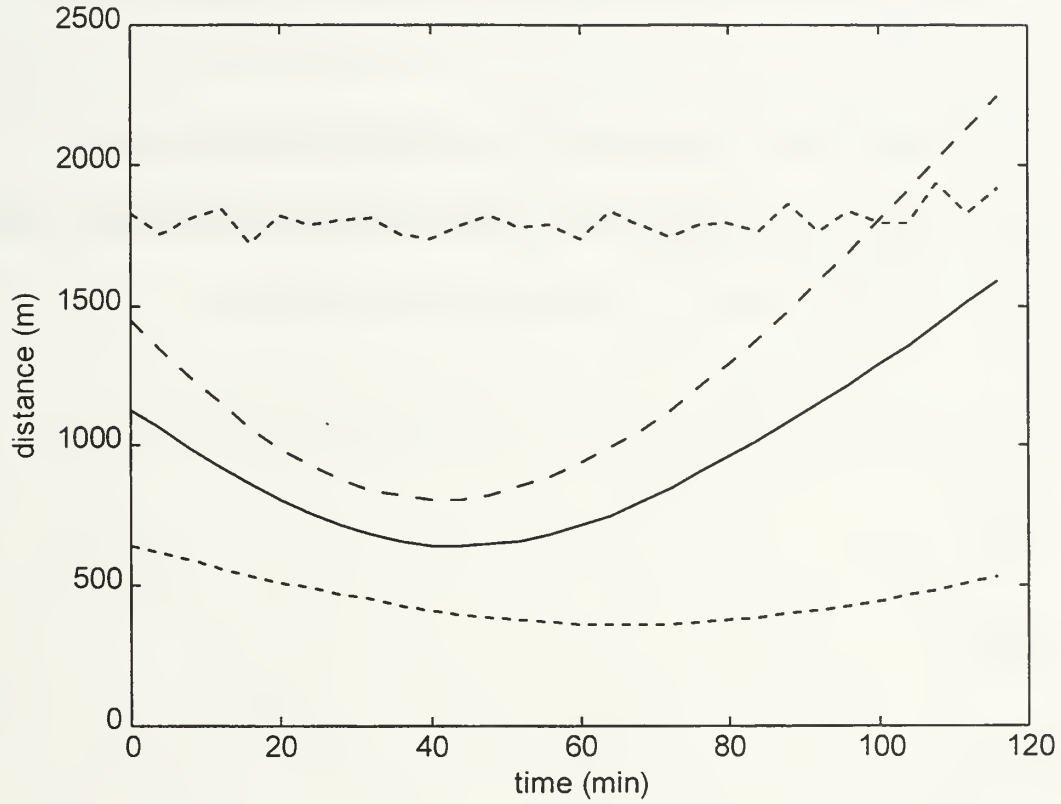

Figure 5.18. Attribute Data Comparison (very high clutter)

The solid line represents the algorithm with the use of attribute data, and the broken line is the algorithm without it. For this simulation an attribute value of $a = \sqrt{10}$ was necessary in order to get a noticeable improvement from the algorithm. As was described in Chapter III, this means that this value of a was used for the target measurements, and a value of $a = 1$ was used for the clutter measurements. For values

less than $a = \sqrt{10}$, no clear advantage could be seen. The fact that a 10dB power advantage is needed for the target data in order to show an improvement is not very encouraging at this point. Given the findings of this research, a large amplitude separation would be necessary to produce any sort of real advantage using the attribute data.

Attribute data was also utilized to see if it could lower the requirements for initialization. Figure 5.19 shows the effect of varying the initialization constant N, which is in N-of-N. A 10dB power ratio is used throughout these simulations.

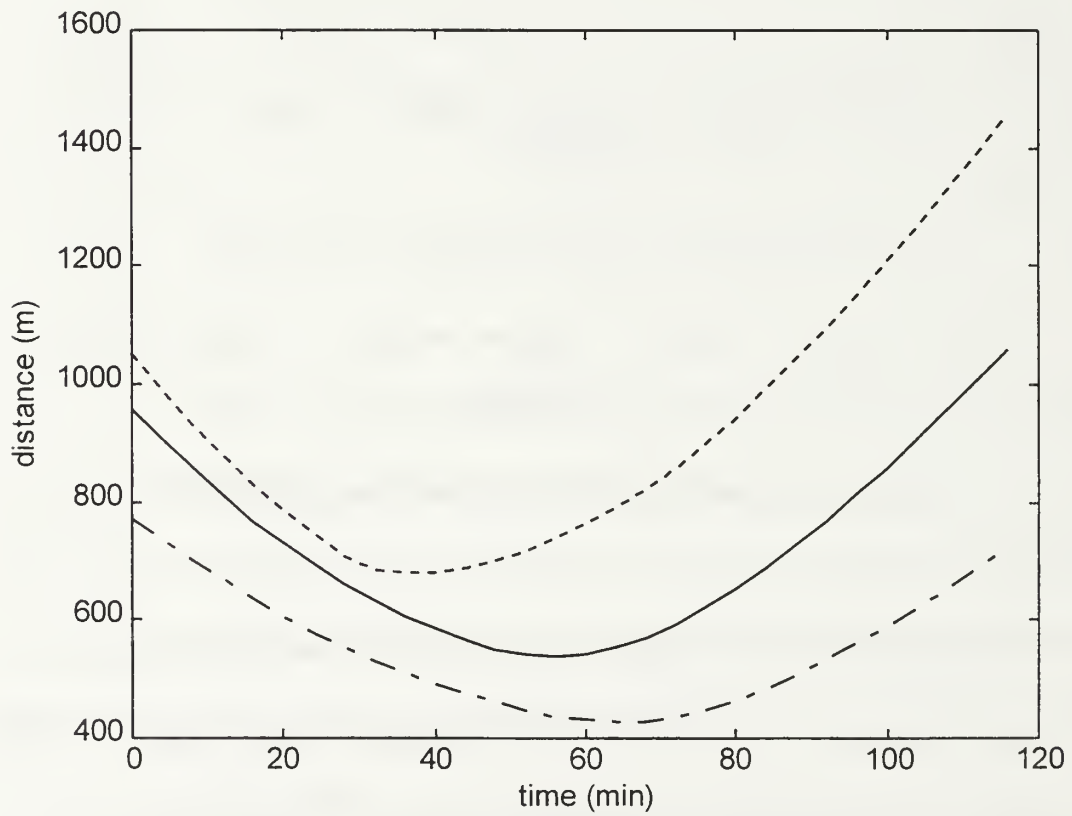

Figure 5.19. Mean Distance Errors with Attribute Data and Varying N

The solid line is the standard PMHT with no attribute data and N=5. The dotted line is the algorithm with attribute data and N=3, and the dash-dot line represents attribute data with N=5. The clutter density for these simulations is a high clutter density of 3.33×10^{-2} clutter points per square kilometer. Here the attribute data with N=3 is not as good as no attribute processing and N=5. However, attribute data with N=5 shows superior results. Therefore, at higher clutter levels, it is not possible to reduce N and make up the difference with 10dB of attribute information.

VI. CONCLUSIONS

A. SUMMARY

This research has explored the possible implementation and initialization of the PMHT algorithm. It has solved and improved many of the aspects of running the algorithm. This includes the development of an initialization routine, a clutter weight, a cut-off for very small value weights, processing in five-scan batches, a maximum allowable velocity of the initial state estimate, clutter weight model inflation, the use of the Extended Kalman smoother, and the use of attribute data.

The results from this research have shown that the PMHT algorithm is a viable player in the data association and tracking arena. It has been shown to outperform the PDAF in all of the scenarios studied here. Furthermore, it has proven to be superior to the MHT in low clutter densities, although it is not as good in the high clutter densities. The PMHT has also shown that it can track a target through a minor turn. Even though it will not track through a radical target maneuver, this is not a glaring weakness since most algorithms require special processing to track a target through a turn, as was discussed in Chapter V.

Presently, the algorithm's greatest shortcoming is in the area of initialization. The requirement for five measurements to line up (N=5) is stringent, especially when the probability of detection is less than one. Unfortunately, the PMHT algorithm has proven to be quite sensitive to the initial estimates. In addition, the algorithm is also easily

modified to make use of attribute data. However, the current 10dB signal to noise ratio is rather high, and probably is not obtainable in the underwater sonar world.

B. FURTHER RESEARCH

The PMHT algorithm has developed quickly since it was proposed by Streit and Luginbuhl in 1995. However, there are still several areas in which improvements and further research need to be addressed. Clearly, the initialization problem needs development, particularly in de-sensitizing the algorithm to the initial estimates.

Another area for further research is the use of attribute data. The current use offers some promise, but more probability research needs to be done with the Rayleigh distribution in order to lower the 10dB signal to noise ratio. Furthermore, using attribute data during initialization needs to be studied more.

The final area where more research is needed is the processing of the algorithm during radical turns and maneuvers. This has been done successfully with other tracking algorithms, but actual simulations with the PMHT algorithm in linking tracks together would be useful. Investigation of the requirements necessary for the new estimates after the turn would be important.

APPENDIX. MATLAB CODE

This appendix contains the code which was used in Matlab 4.2c[1] to simulate the PMHT algorithm. The first set of code is for the two crossing target scenario. The second set of code is for a straight track with attribute data. For the other scenarios, slight modifications were made to either of these programs.

Crossing Tracks

```
%Probabilistic Multi-Hypothesis Tracking (version 36)
% thesis by Capt. Darin T. Dunham, USMC
% advisor:  Prof. R. Gary Hutchins, NPS
%
% two tracks with clutter--
%   first two meas are one std error,
%   the other meas are uniform clutter.
%
% Uses clutter tracking model.
%
% Tracks in Tstep scan increments.
%
% Uses attribute data for each measurement      NOT IMPLEMENTED
%  which is a random Rayleigh distribution.
%
% Tracks cross.
%
% Uses first 5 points to initialize.
%
% Uses an Extended Kalman instead of debiased eqns.
%
% Computes the mean track to show bias of second track.
%
```

[1] Matlab® copyright© 1984-94 The MathWorks, Inc., All Rights Reserved, Version 4.2c, Nov. 23, 1994. The MathWorks, Inc., 24 Prime Park Way, Natick, MA 01760.

```
clear
tic

%Constants
M=3;                    %three models, two track, one clutter
N=5;                    %number of points used to initialize
Nt=2;                   %number of tracks
Nc=10;                  %number of clutter points
T=30;                   %number of scans
Tstep=5;                %algorithm increment step size (change cwt)
dt=4;                   %in minutes
I=100;                  %max number of iterations
sr=100;                 %std for range
sth=pi/60;              %std for bearing
s=1;                    %std for attribute on a true target
sc=1;                   %std for attribute on clutter
q=1;                    %Q coefficient
maxv1=308.66667;        %max allowable initial velocity for track 1 (10 knots)
maxv2=308.66667;        %max allowable initial velocity for track 2 (10 knots)
thr=10.5966;            %threshold for 3 std
stop=1e-6;              %convergence stopping parameter
J=500;                  %number of loops thru the simulation

%Constant vectors
cwt=1e-10*[1e-2 1e-2 1 1 1 1];          %clutter model weight

%Initiation for actual track (meters)
X1init=[30200, 92.6, 30400, -123.46667]';
X2init=[30200, 92.6, 16078, 123.46667]';
Y1=zeros(4,T);
Y2=zeros(4,T);
Y1(:,1)=X1init;
Y2(:,1)=X2init;
Q=q*[(dt^3)/3 (dt^2)/2 0 0; (dt^2)/2 dt 0 0;
  0 0 (dt^3)/3 (dt^2)/2; 0 0 (dt^2)/2 dt];
R=[sr^2 0; 0 sth^2];
A=[0 1 0 0; 0 0 0 0; 0 0 0 1; 0 0 0 0];
C=[1 0 0 0; 0 0 1 0];
Phi=eye(4) + A*dt;
invPhi=eye(4) - A*dt;
for t=1:T-1,
  Y1(:,t+1)=Phi*Y1(:,t);
  Y2(:,t+1)=Phi*Y2(:,t);
```

```
end
Y1pol=xy2polar(C*Y1);
Y2pol=xy2polar(C*Y2);

%Initialize error vectors
errm=zeros(1,T);
erre=zeros(1,T);
errm2=zeros(1,T);
erre2=zeros(1,T);
count1=0;
count2=0;

%initialize mean estimated track vectors
X1m=zeros(4,T);
X2m=zeros(4,T);

%Loop through simulation J times**************************************
for j=1:J,
j

%Initialize tracker
p1=0.2*ones(1,T);        %p1(t)-prob that target 1 has a meas in time t
p2=0.2*ones(1,T);
p3=1.0*ones(1,T);

%Generate range and bearing measurements
Z1pol=Y1pol + sqrt(R)*randn(2,T);
Z2pol=Y2pol + sqrt(R)*randn(2,T);

%Convert measurements to cartesian using Debiased Eqns.
mu=1 - (exp(-sth^2) - exp(-(sth^2)/2));
Z1=(mu*eye(2))*polar2xy(Z1pol);
Z2=(mu*eye(2))*polar2xy(Z2pol);
Rs=[convert(Z1pol,sr,sth); convert(Z2pol,sr,sth)];

%Generate attribute data for targets
Z=[Z1pol; raylrnd(s,1,T); Z2pol; raylrnd(s,1,T)];

%Generate clutter measurements
for m=1:Nc,
  Zipol=xy2polar([1.5e4*rand(1,T)+2.8e4; 2e4*rand(1,T)+1.5e4]);
  Z=[Z; Zipol; raylrnd(sc,1,T)];
end
```

```
%Build Zbar
Zbar1=Z(1,:);
Zbar2=Z(2,:);
for v=4:3:3*(Nt+Nc),
  Zbar1=[Zbar1; Z(v,:)];
  Zbar2=[Zbar2; Z(v+1,:)];
end

%Initialize X1 and X2
X1=zeros(4,T);
F=[C; C*Phi; C*(Phi^2); C*(Phi^3); C*(Phi^4)];
R1=form(Rs(1:3,1));
R2=form(Rs(1:3,2));
R3=form(Rs(1:3,3));
R4=form(Rs(1:3,4));
R5=form(Rs(1:3,5));
Sig=[R1 zeros(2,8); zeros(2,2) R2 zeros(2,6);
    zeros(2,4) R3 zeros(2,4); zeros(2,6) R4 zeros(2,2); zeros(2,8) R5];
invSig=inv(Sig);
K=inv(F'*invSig*F)*F'*invSig;
Zbar=[Z1(:,1); Z1(:,2); Z1(:,3); Z1(:,4); Z1(:,5)];
X1(:,1)=K*Zbar;
Pv1(:,2)=reshape((K*Sig*K'),16,1);

X2=zeros(4,T);
R1=form(Rs(4:6,1));
R2=form(Rs(4:6,2));
R3=form(Rs(4:6,3));
R4=form(Rs(4:6,4));
R5=form(Rs(4:6,5));
Sig=[R1 zeros(2,8); zeros(2,2) R2 zeros(2,6);
       zeros(2,4) R3 zeros(2,4); zeros(2,6) R4 zeros(2,2); zeros(2,8) R5];
invSig=inv(Sig);
K=inv(F'*invSig*F)*F'*invSig;
Zbar=[Z2(:,1); Z2(:,2); Z2(:,3); Z2(:,4); Z2(:,5)];
X2(:,1)=K*Zbar;
Pv2(:,2)=reshape((K*Sig*K'),16,1);

%Limit initial velocity
initvel1=sqrt(X1(2,1)^2 + X1(4,1)^2);
initvel2=sqrt(X2(2,1)^2 + X2(4,1)^2);
vfactor1=initvel1/maxv1;
```

```
vfactor2=initvel2/maxv2;
if vfactor1>1,
  X1(2,1)=X1(2,1)/vfactor1;
  X1(4,1)=X1(4,1)/vfactor1;
end
if vfactor2>1,
  X2(2,1)=X2(2,1)/vfactor2;
  X2(4,1)=X2(4,1)/vfactor2;
end

%Predict initial estimate for first five points
for t=2:5,
  X1(:,t)=Phi*X1(:,t-1);
  X2(:,t)=Phi*X2(:,t-1);
  Pv1(:,2*t)=reshape((Phi*formP(Pv1(:,2*(t-1)))*Phi' + Q),16,1);
  Pv2(:,2*t)=reshape((Phi*formP(Pv2(:,2*(t-1)))*Phi' + Q),16,1);
end
U1=X1(:,1:5);
U2=X2(:,1:5);

%Track in five scan increments**********************************
for Ti=5:Tstep:T,

  %Begin iterations
  for i=1:I,

    %store last target meas prob
    p1p=p1;
    p2p=p2;
    p3p=p3;

    for t=1:Ti,
      n(t)=Nt+Nc;

     %compute weights
      for r=1:n(t),
        zt=Z((3*r-2):(3*r-1),t)-xy2polar(C*X1(:,t));
        Ck=Cekf(X1(:,t));
        Sig=Ck*reshape(Pv1(:,2*t),4,4)*Ck' + R;
        den=2*pi*sqrt(det(Sig));
        w1(r,t)=exp(-0.5*zt'*inv(Sig)*zt)/den;
        if zt'*inv(Sig)*zt > thr,
          w1(r,t)=1e-20;
```

```
    end

    zt=Z((3*r-2):(3*r-1),t)-xy2polar(C*X2(:,t));
    Ck=Cekf(X2(:,t));
    Sig=Ck*reshape(Pv2(:,2*t),4,4)*Ck' + R;
    den=2*pi*sqrt(det(Sig));
    w2(r,t)=exp(-0.5*zt'*inv(Sig)*zt)/den;
    if zt'*inv(Sig)*zt > thr,
      w2(r,t)=1e-20;
    end

    w3(r,t)=cwt(Ti/Tstep);
    sum1=(p1(t)*w1(r,t)+p2(t)*w2(r,t)+p3(t)*w3(r,t));
    w1(r,t)=w1(r,t)/sum1;
    w2(r,t)=w2(r,t)/sum1;
    w3(r,t)=w3(r,t)/sum1;
  end

  %compute mean meas weight for target m at time t
  w1m(t)=(1/n(t))*sum(w1(:,t));
  w2m(t)=(1/n(t))*sum(w2(:,t));
  w3m(t)=(1/n(t))*sum(w3(:,t));

  %update target meas prob
  p1(t)=w1m(t)*p1(t);
  p2(t)=w2m(t)*p2(t);
  p3(t)=w3m(t)*p3(t);

  %compute target meas centroid
  W1=w1(:,t)/(n(t)*w1m(t));
  W2=w2(:,t)/(n(t)*w2m(t));
  Z1hat(:,t)=[W1'*Zbar1(:,t); W1'*Zbar2(:,t)];
  Z2hat(:,t)=[W2'*Zbar1(:,t); W2'*Zbar2(:,t)];

end

%run Extended Kalman smoother
y1hat(:,1)=X1(:,1);
y2hat(:,1)=X2(:,1);

%forward recursion
for t=1:Ti-1,
  Pt=Phi*reshape(Pv1(:,2*t),4,4)*Phi' + Q;
```

```
    Pv1(:,2*t+1)=Pt(:);
    y1hat(:,t+1)=Phi*y1hat(:,t);
    k=n(t+1)*p1(t+1);
    Ck=Cekf(y1hat(:,t+1));
    G=k*Pt*Ck'*inv(k*Ck*Pt*Ck' + R);
    Pv1(:,2*t+2)=reshape((Pt-G*Ck*Pt),16,1);
    y1hat(:,t+1)=y1hat(:,t+1)+G*(Z1hat(:,t+1)-xy2polar(C*y1hat(:,t+1)));

    Pt=Phi*reshape(Pv2(:,2*t),4,4)*Phi' + Q;
    Pv2(:,2*t+1)=Pt(:);
    y2hat(:,t+1)=Phi*y2hat(:,t);
    k=n(t+1)*p2(t+1);
    Ck=Cekf(y2hat(:,t+1));
    G=k*Pt*Ck'*inv(k*Ck*Pt*Ck' + R);
    Pv2(:,2*t+2)=reshape((Pt-G*Ck*Pt),16,1);
    y2hat(:,t+1)=y2hat(:,t+1)+G*(Z2hat(:,t+1)-xy2polar(C*y2hat(:,t+1)));
  end
  X1(:,Ti)=y1hat(:,Ti);
  X2(:,Ti)=y2hat(:,Ti);

  %backward recursion
  for t=Ti-1:-1:1,
    Ptt=reshape(Pv1(:,2*t),4,4);
    invPt=inv(reshape(Pv1(:,2*t+1),4,4));
    X1(:,t)=y1hat(:,t)+Ptt*Phi'*invPt*(X1(:,t+1)-Phi*y1hat(:,t));

    Ptt=reshape(Pv2(:,2*t),4,4);
    invPt=inv(reshape(Pv2(:,2*t+1),4,4));
    X2(:,t)=y2hat(:,t)+Ptt*Phi'*invPt*(X2(:,t+1)-Phi*y2hat(:,t));
  end

  %check for convergence
  diff=sum(abs(p1-p1p))+sum(abs(p2-p2p))+sum(abs(p3-p3p));
  if diff<stop,
    break
  end

end
i

%plot track output
figure(1)
 plot(Y1(1,1:Ti),Y1(3,1:Ti),X1(1,1:Ti),X1(3,1:Ti),Y2(1,1:Ti),...
```

```
        Y2(3,1:Ti),X2(1,1:Ti),X2(3,1:Ti))
   axis('equal'), title('Tracking Algorithm Output')
   xlabel('x (meters)'), ylabel('y (meters)')

  %Predict for next Tstep points if necessary
  if Ti<T,
    mid=floor(Ti/2);
    initvel1=sqrt(X1(2,mid)^2 + X1(4,mid)^2);
    initvel2=sqrt(X2(2,mid)^2 + X2(4,mid)^2);
    vfactor1=initvel1/maxv1;
    vfactor2=initvel2/maxv2;
    if vfactor1>1,
      X1(2,mid)=X1(2,mid)/vfactor1;
      X1(4,mid)=X1(4,mid)/vfactor1;
    end
    if vfactor2>1,
      X2(2,mid)=X2(2,mid)/vfactor2;
      X2(4,mid)=X2(4,mid)/vfactor2;
    end
    for t=mid+1:Ti+Tstep,
      X1(:,t)=Phi*X1(:,t-1);
      Pv1(:,2*t)=reshape((Phi*reshape(Pv1(:,2*(t-1)),4,4)*Phi'+Q),16,1);
      X2(:,t)=Phi*X2(:,t-1);
      Pv2(:,2*t)=reshape((Phi*reshape(Pv2(:,2*(t-1)),4,4)*Phi'+Q),16,1);
    end
    for t=mid-1:-1:1,
      X1(:,t)=invPhi*X1(:,t+1);
      X2(:,t)=invPhi*X2(:,t+1);
    end
  end

end

 %update measurement & estimate errors for track 1
 errx=Y1(1,:)-Z1(1,:);
 erry=Y1(3,:)-Z1(2,:);
 errm=sqrt(errx.^2 + erry.^2)./J + errm;
 errx=Y1(1,:)-X1(1,:);
 erry=Y1(3,:)-X1(3,:);
 errej=sqrt(errx.^2 + erry.^2);
 erre=errej./J + erre;
 if max(errej) > 2000,
    count1=count1+1;
```

```
end

%update measurement & estimate errors for track 2
errx=Y2(1,:)-Z2(1,:);
erry=Y2(3,:)-Z2(2,:);
errm2=sqrt(errx.^2 + erry.^2)./J + errm2;
errx=Y2(1,:)-X2(1,:);
erry=Y2(3,:)-X2(3,:);
errej=sqrt(errx.^2 + erry.^2);
erre2=errej./J + erre2;
if max(errej) > 2000,
  count2=count2+1;
end

%update mean estimated tracks
X1m=X1m + X1./J;
X2m=X2m + X2./J;

end
%End J times loop*****************************************************

%plot errors
figure(2)
 load kserror
 plot(1:T,errm,1:T,erre,1:T,errm2,1:T,erre2,1:T,err1(1:T),1:T,err2(1:T))
 title('Measurement Error & Estimate Error, over 100 runs')
 xlabel('time'), ylabel('distance (m)')

figure(3)
 plot(Y1(1,1:T),Y1(3,1:T),X1m(1,1:T),X1m(3,1:T),Y2(1,1:T),...
Y2(3,1:T),X2m(1,1:T),X2m(3,1:T))
 axis('equal'), title('Tracking Algorithm Output')
 xlabel('x (meters)'), ylabel('y (meters)')

time=toc/60;
[time count1 count2]
```

Attribute Data

```
%Probabilistic Multi-Hypothesis Tracking (version 35)
% thesis by Capt. Darin T. Dunham, USMC
```

```
% advisor:  Prof. R. Gary Hutchins, NPS
%
% one track with clutter--
%   first meas is one std error,
%   the other meas are uniform clutter.
%
% Uses clutter tracking model.
%
% Tracks in Tstep scan increments.
%
% Uses attribute data for each measurement
%  which is a random Rayleigh distribution.
%
% Uses first 5 points to initialize.
%
% Uses an Extended Kalman instead of debiased eqns.
%

%clear
tic

%Constants
M=2;                    %two models
N=5;                    %number of points used to initialize
Nt=1;                   %number of tracks
Nc=20;                  %number of clutter points
T=30;                   %number of scans
Tstep=5;                %algorithm increment step size
dt=4;                   %in minutes
I=100;                  %max number of iterations
sr=100;                 %std for range
sth=pi/60;              %std for bearing
%s=sqrt(10);            %std for attribute on a true target
sc=1;                   %std for attribute on clutter
q=1;                    %Q coefficient
maxv=308.66667;         %max allowable predict velocity (10 knots)
thr=10.5966;            %threshold for 3 std
stop=1e-8;              %convergence value for Pi
J=500;                  %number of loops thru the simulation

%Constant vectors
cwt=1e-10*[1e-2 1e-2 1 1 1 1];   %clutter model weight
```

```
%initiation for actual track (meters)
X1init=[30200, 92.6, 30400, -123.46667]';
%X1init=[30200, 92.6, 16078, 123.46667]';
Y1=zeros(4,T);
Y1(:,1)=X1init;
C=[1 0 0 0; 0 0 1 0];
Q=q*[(dt^3)/3 (dt^2)/2 0 0; (dt^2)/2 dt 0 0;
  0 0 (dt^3)/3 (dt^2)/2; 0 0 (dt^2)/2 dt];
R=[sr^2 0; 0 sth^2];
A=[0 1 0 0; 0 0 0 0; 0 0 0 1; 0 0 0 0];
Phi=eye(4) + A*dt;
invPhi=eye(4) - A*dt;
for t=1:T-1,
  Y1(:,t+1)=Phi*Y1(:,t);
end
Y1pol=xy2polar(C*Y1);
errm=zeros(1,T);
erre=zeros(1,T);
count=0;

%Loop through simulation J times**************************************
for j=1:J,
j

%Initialize tracker
p1=0.2*ones(1,T);   %p1(t)-prob that tar 1 has a meas in time t
p2=1.0*ones(1,T);

%Generate range and bearing measurements
Z1pol=Y1pol + sqrt(R)*randn(2,T);

%Convert measurements to cartesian using Debiased Eqns.
mu=1 - (exp(-sth^2) - exp(-(sth^2)/2));
Z1=(mu*eye(2))*(polar2xy(Z1pol));
Rs=convert(Z1pol,sr,sth);

%Generate attribute data for target & measurement covariance
Z=[Z1pol; raylrnd(s,1,T)];

%Generate clutter measurements
for m=1:Nc,
  Zipol=xy2polar([1.5e4*rand(1,T)+2.8e4; 2e4*rand(1,T)+1.5e4]);
  Z=[Z; Zipol; raylrnd(sc,1,T)];
```

```
end

%Build Zbar
Zbar1=Z(1,:);
Zbar2=Z(2,:);
for v=4:3:3*(Nt+Nc),
  Zbar1=[Zbar1; Z(v,:)];
  Zbar2=[Zbar2; Z(v+1,:)];
end

%Initialize X
X1=zeros(4,T);
F=[C; C*Phi; C*(Phi^2); C*(Phi^3); C*(Phi^4)];
R1=form(Rs(:,1));
R2=form(Rs(:,2));
R3=form(Rs(:,3));
R4=form(Rs(:,4));
R5=form(Rs(:,5));
Sig=[R1 zeros(2,8); zeros(2,2) R2 zeros(2,6);
    zeros(2,4) R3 zeros(2,4); zeros(2,6) R4 zeros(2,2); zeros(2,8) R5];
invSig=inv(Sig);
K=inv(F'*invSig*F)*F'*invSig;
Zbar=[Z1(:,1); Z1(:,2); Z1(:,3); Z1(:,4); Z1(:,5)];
X1(:,1)=K*Zbar;
Pv(:,2)=reshape((K*Sig*K'),16,1);

%Limit initial velocity
initvel=sqrt(X1(2,1)^2 + X1(4,1)^2);
vfactor=initvel/maxv;
if vfactor>1,
  X1(2,1)=X1(2,1)/vfactor;
  X1(4,1)=X1(4,1)/vfactor;
end

%Predict initial estimate for first five points
for t=2:5,
  X1(:,t)=Phi*X1(:,t-1);
  Pv(:,2*t)=reshape((Phi*formP(Pv(:,2*(t-1)))*Phi' + Q),16,1);
end
U1=X1(:,1:5);

%Track in Tstep scan increments**********************************
for Ti=5:Tstep:T,
```

```
%Begin iterations
for i=1:I,

  %store last target meas prob
  p1p=p1;
  p2p=p2;

  for t=1:Ti,
    n(t)=Nt+Nc;

    %compute weights
    for r=1:n(t),
      zt=Z((3*r-2):(3*r-1),t)-xy2polar(C*X1(:,t));
      Ck=Cekf(X1(:,t));
      Sig=Ck*reshape(Pv(:,2*t),4,4)*Ck' + R;
      den=2*pi*sqrt(det(Sig));
      w1(r,t)=raylpdf(Z(3*r,t),s)*exp(-0.5*zt'*inv(Sig)*zt)/den;
      if zt'*inv(Sig)*zt > thr,
        w1(r,t)=1e-20;
      end
      w2(r,t)=raylpdf(Z(3*r,t),sc)*cwt(Ti/Tstep);
      sum1=(p1(t)*w1(r,t)+p2(t)*w2(r,t));
      w1(r,t)=w1(r,t)/sum1;
      w2(r,t)=w2(r,t)/sum1;
    end

    %compute mean meas weight for target m at time t
    w1m(t)=(1/n(t))*sum(w1(:,t));
    w2m(t)=(1/n(t))*sum(w2(:,t));

    %update target meas prob
    p1(t)=w1m(t)*p1(t);
    p2(t)=w2m(t)*p2(t);

    %compute target meas centroid
    W=w1(:,t)/(n(t)*w1m(t));
    Zhat(:,t)=[W'*Zbar1(:,t); W'*Zbar2(:,t)];

  end

  %run Extended Kalman smoother
  yhat(:,1)=X1(:,1);
```

```
  %forward recursion
  for t=1:Ti-1,
    Pt=Phi*reshape(Pv(:,2*t),4,4)*Phi' + Q;
    Pv(:,2*t+1)=Pt(:);
    yhat(:,t+1)=Phi*yhat(:,t);
    k=n(t+1)*p1(t+1);
    Ck=Cekf(yhat(:,t+1));
    G=k*Pt*Ck'*inv(k*Ck*Pt*Ck' + R);
    Pv(:,2*t+2)=reshape((Pt-G*Ck*Pt),16,1);
    yhat(:,t+1)=yhat(:,t+1)+G*(Zhat(:,t+1)-xy2polar(C*yhat(:,t+1)));
  end
  X1(:,Ti)=yhat(:,Ti);

  %backward recursion
  for t=Ti-1:-1:1,
    Ptt=reshape(Pv(:,2*t),4,4);
    invPt=inv(reshape(Pv(:,2*t+1),4,4));
    X1(:,t)=yhat(:,t)+Ptt*Phi'*invPt*(X1(:,t+1)-Phi*yhat(:,t));
  end

  %check for convergence
  diff=sum(abs(p1-p1p))+sum(abs(p2-p2p));
  if diff < stop,
    break
  end

end
U2=X1(:,1:Ti);
i

%plot track output
figure(1)
 plot(Y1(1,1:Ti),Y1(3,1:Ti),X1(1,1:Ti),X1(3,1:Ti))
 title('Tracking Algorithm Output')
 xlabel('x (meters)'), ylabel('y (meters)')

%Predict for next five points if necessary
if Ti<T,
  mid=floor(Ti/2);
  initvel=sqrt(X1(2,mid)^2 + X1(4,mid)^2);
  vfactor=initvel/maxv;
  if vfactor>1,
```

```
      X1(2,mid)=X1(2,mid)/vfactor;
      X1(4,mid)=X1(4,mid)/vfactor;
    end
    for t=mid+1:Ti+Tstep,
      X1(:,t)=Phi*X1(:,t-1);
      Pv(:,2*t)=reshape((Phi*reshape(Pv(:,2*(t-1)),4,4)*Phi' + Q),16,1);
    end
    for t=mid-1:-1:1,
      X1(:,t)=invPhi*X1(:,t+1);
    end
  end

end

 errx=Y1(1,:)-Z1(1,:);
 erry=Y1(3,:)-Z1(2,:);
 errm=sqrt(errx.^2 + erry.^2)./J + errm;
 errx=Y1(1,:)-X1(1,:);
 erry=Y1(3,:)-X1(3,:);
 errej=sqrt(errx.^2 + erry.^2);
 erre=errej./J + erre;
 if max(errej) > 2000,
   count=count+1;
 end

end
%End J times loop*****************************************************

%plot errors
figure(2)
 load kserror
 plot(1:T,erre,1:T,err1(1:T),1:T,err2(1:T))
 title('Measurement Error & Estimate Error, over 100 runs')
 xlabel('time'), ylabel('distance (m)')

vel=sqrt(U1(2,1)^2 + U1(4,1)^2);
anglediff=(180/pi)*(atan2(U1(4,1),U1(2,1))-atan2(Y1(4,1),Y1(2,1)));
time=toc/60;
[vel anglediff time count]
```

LIST OF REFERENCES

[1] Streit, Roy L. and Luginbuhl, Todd E., *Probabilistic Multi-Hypothesis Tracking*, NUWC-NPT Technical Report 10,428, 15 February 1995.

[2] Hutchins, Robert G. and Dunham, Darin T., "Evaluation of a Probabilistic Multihypothesis Tracking Algorithm in Cluttered Environments," *Proceedings from The Thirtieth Asilomar Conference on Signals, Systems, and Computers*, November 1996.

[3] Lerro, Don and Bar-Shalom, Yaakov, "Tracking with Debiased Consistent Converted Measurements Versus EKF," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 29, No. 3, July 1993.

[4] Bar-Shalom, Yaakov and Li, Xiao-Rong, *Multitarget-Multisensor Tracking: Principles and Techniques*, Yaakov Bar-Shalom, Storrs, CT, 1995.

[5] Blackman, Samuel S., *Multiple-Target Tracking with Radar Applications*, Artech House, Norwood, MA, 1986.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center 8725 John J. Kingman Rd., STE 0944 Ft. Belvoir, VA 22060-6218	2
2.	Dudley Knox Library Naval Postgraduate School 411 Dyer Rd. Monterey, CA 93943-5101	2
3.	Director, Training and Education MCCDC, Code C46 1019 Elliot Road Quantico, VA 22134-5027	1
4.	Director, Marine Corps Research Center MCCDC, Code C40RC 2040 Broadway Street Quantico, VA 22134-5107	2
5.	Director, Studies and Analysis Division MCCDC, Code C45 300 Russell Road Quantico, VA 22134-5130	1
6.	Chairman, Code EC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	1
7.	Professor Robert G. Hutchins, Code EC/Hu Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	1
8.	Professor Harold A. Titus, Code EC/Ts Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	1

9.	Dr. Michael Shields, Code EC/Sl Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	1
10.	Captain Darin T. Dunham 250 Apple Blossom Road Pataskala, OH 43062-9115	2
11.	Dr. Roy L. Streit, Code 2214 Naval Undersea Warfare Center Newport, RI 02841-5047	1